Departamento de Ingeniería Mecánica

PROYECTO FIN DE CARRERA

DESARROLLO DE UNA LIBRERÍA MULTICUERPO CON ECOSIMPRO

Autor: D.Alejandro Sánchez González

Directores: Dr.Dña.María Jesús López Boada

Dr.Dña.Ester Olmeda Santamaría

Titulación: Ingeniería Técnica Industrial, Mecánica

Leganés, diciembre de 2011

Título: Desarrollo de una librería Multicuerpo con EcosimPro
Autor: D.Alejandro Sánchez González
Directores: Dr.Dña.María Jesús López Boada
Dr.Dña.Ester Olmeda Santamaría

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 22 de diciembre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Resumen

En el presente Proyecto de Fin de Carrera se analiza la filosofía de modelado de sistemas multicuerpo, y se estudia como llevar a cabo este tipo de programación en una herramienta de modelado y simulación no-causal llamada EcosimPro.

En este documento se explica qué es y como se trabaja con la programación multicuerpo. Por otro lado, se analiza el funcionamiento del programa Ecosimpro y se compara con otro programa similar llamado Modelica.

Por último, se crea una librería de programación multicuerpo en EcosimPro, y posteriormente se simula un experimento tomado de Modelica para poder comparar los resultados.

Abstract

This Project examines the philosophy of multibody systems modeling, and considers how to perform this type of programming with a non-casual modeling tool called EcosimPro.

This document explains how to work with multibody programming. On the other hand, analizes how the program EcosimPro works and is compared with another similar program called Modelica.

Finally, it has created a programming multibody library in EcosimPro, subsequently, it has simulated an experiment taken from Modelica to compare the results.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

En la actualidad nadie pone en duda la gran utilidad de la simulación para el diseño y desarrollo de nuevos productos. Es una técnica fundamental en varias disciplinas para mejorar los diseños o recortar costes. Las técnicas de simulación se llevan usando durante muchos años en diversos campos como el aeroespacial, la automoción, la industria química, etc.

La disciplina de la simulación ha evolucionado mucho en los últimos años, y en la actualidad existen herramientas de simulación con una potencia y versatilidad impensables hace poco tiempo. Estas herramientas permiten modelar sistemas con un grado de precisión muy alto, y esto unido al aumento de la potencia de los ordenadores hace que se pueda hacer un diseño previo de los productos antes de fabricarlos.

EcosimPro es una herramienta de modelado y simulación para sistemas multidisciplinares basados en ecuaciones algebraico-diferenciales y eventos discretos. Este programa está siendo utilizado por numerosas empresas y centros de investigación, destacando su elección como herramienta oficial de la Agencia Espacial Europea para la simulación de sistemas de propulsión espacial.

EcosimPro se enmarca dentro del área de las herramientas de simulación 1D, es válida para modelar sistemas en los que la geometría se ha simplificado y los componentes se intercambian flujos en 1D.

Al no trabajar EcosimPro con sistemas de varios cuerpos, surgío la idea de intentar desarrollar una librería en EcosimPro que permitiese el modelado de sistemas multicuerpo con tres dimensiones.

1.2 Objetivos

El objetivo principal que se pretende conseguir con el presente Proyecto Fin de Carrera es la creación de una librería en EcosimPro que permita iniciarse en el modelado de sistemas multicuerpo. Para ello, se va a simular en EcosimPro un experimento de sistema multicuerpo tomado del programa Modelica, y así poder comprobar con este programa que los resultados obtenidos son correctos.

Para la modelación de sistemas multicuerpo con EcosimPro se han seguido los siguientes pasos:

- Análisis de la programación multicuerpo con Modelica.
- Estudio del modelado con EcosimPro.
- Modelado del sistema multicuerpo con EcosimPro.
- Comparativa del los resultados obtenidos con los de Modelica.
- Conclusiones tras la evaluación de los resultados.

1.3 Estructura de la memoria

El presente proyecto consta de siete capítulos siendo el primero de ellos la presente Introducción.

En el capítulo 2 “Sistemas Multicuerpo”, se explica el concepto de este tipo se sistemas, analizando los distintos tipos de fuerzas, juntas y restricciones que forman la estructura de dichos sistemas.

En el capítulo 3 “Modelado”, se exponen los conceptos básicos del modelado orientado a objetos y se detallan las características principales del modelado con EcosimPro.

En el capítulo 4 “Creación de librerías multicuerpo”, se muestran y explican las librerías multicuerpo desarrolladas en EcosimPro.

En el capítulo 5 “Resultados de la simulación”, se muestran los resultados obtenidos tras la simulación sobre un modelo formado por un péndulo y un amortiguador. En este apartado, se comprueba que los resultados obtenidos en la simulación con EcosimPro son los mismos que se obtuvieron con Modelica.

En el capítulo 6 “Conclusiones y desarrollos futuros”, se presentan las principales conclusiones obtenidas durante la realización de la simulación en el sistema del péndulo y la posibilidad de ampliar y mejorar la librería multicuerpo creada para EcosimPro.

El último capítulo está dedicado al conjunto de referencias bibliográficas empleadas en este documento.

Capítulo 2

Sistemas Multicuerpo

2.1 Introducción a los Sistemas Multicuerpo

Un sistema multicuerpo es un sistema fundamentalmente mecánico, formado por varios sólidos rígidos parcialmente unidos entre sí mediante pares cinemáticos y elementos de fuerza. Los pares cinemáticos son uniones imperfectas entre sólidos, que permiten algunos grados de libertad y restringen otros.

En la figura 2.1 se observa un ejemplo de un sistema multicuerpo con sus elementos de fuerza y sus uniones.

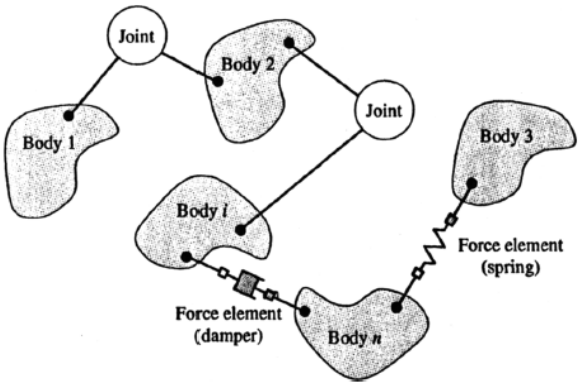

Figura 2.1. Ejemplo de sistema multicuerpo

El concepto de sistema multicuerpo es parecido a una clase de mecanismo, el concepto de mecanismo es algo más general y se utiliza para designar a cualquier tipo de sistema mecánico con elementos móviles como por ejemplo un engranaje. Los sistemas multicuerpo se corresponderían mejor con los mecanismos de barras articuladas [1].

Casi cualquier cosa puede ser modelada como sistema multicuerpo, desde un sistema de estrellas hasta un avión como el de la figura 2.2.

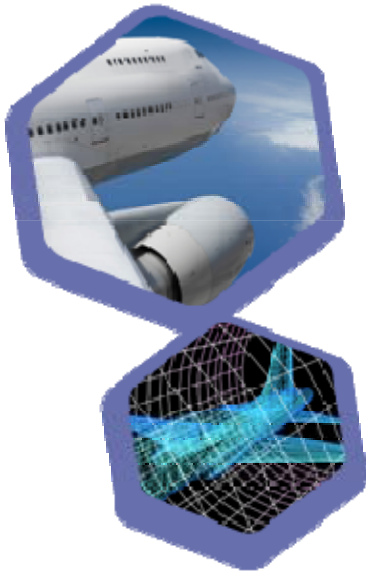

Figura 2.2. Avión modelado con Modelica

Los problemas de sistemas multicuerpo suelen implicar grandes desplazamientos, con importantes cambios en la geometría a lo largo del movimiento. Suelen ser asimismo fuertemente no lineales y presentar

discontinuidades originadas, además de por los citados cambios de geometría, por las fuerzas exteriores aplicadas, por rozamientos de distinta naturaleza, etc.

La simulación de sistemas mecánicos complejos (vehículos, satélites artificiales, etc.) conlleva problemas numéricos largos y difíciles de resolver, por lo que es muy importante disponer de métodos matemáticos que sean robustos y muy eficientes. En la simulación de sistemas multicuerpo es más importante la interactividad e incluso el tiempo real que en los cálculos de elementos finitos. Esto se debe a que para comprender los movimientos de gran amplitud típicos de estos sistemas es fundamental utilizar una escala de tiempos lo más parecida a la real.

Una de las técnicas analíticas más poderosas de la mecánica consiste en aislar una porción del sistema con el fin de estudiar el comportamiento de una de sus partes. Cuando aislamos dicha parte, el efecto que tenía el resto del sistema sobre ella se remplaza por las fuerzas y momentos correspondientes [2].

Aunque estas fuerzas pueden ser efectos internos ejercidos en todo el sistema, son efectos externos cuando se aplican a la parte aislada. Al subsistema aislado que se produce, junto con todas las fuerzas y momentos atribuibles a efectos externos y a las reacciones con el sistema principal se le denomina diagrama de cuerpo libre.

En el caso de los sistemas multicuerpo, se puede simplificar su análisis dinámico de manera apreciable aislando cada uno de los cuerpos que lo componen, y analizándolos mediante el uso de diagramas de cuerpo libre. Cuando todos los elementos han recibido este tratamiento, pueden reunirse los resultados del análisis realizado para generar información concerniente al comportamiento del sistema total [3].

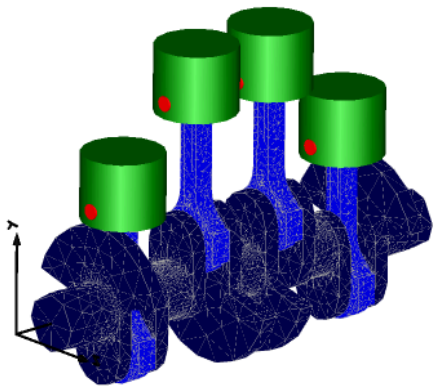

Figura 2.3. Cigüeñal de un motor

Para mecanismos multicuerpo es básico entender el movimiento de subsistemas (cuerpos o componentes). En la figura 2.3 se observa el cigüeñal de un motor, es importante conocer el movimiento de cada componente para diseñar el conjunto. Algunas de las investigaciones más recientes sobre el movimiento de los cuerpos se agrupan en tres campos diferentes: mecánica del cuerpo rígido, mecánica estructural y mecánica continúa [4]:

- El término de **mecánica del cuerpo rígido** implica que la deformación del cuerpo es baja, considerando que son pequeñas estas deformaciones y que no afectan al movimiento del cuerpo. El movimiento de un cuerpo rígido en el espacio puede ser completamente descrito usando seis coordenadas generalizadas.

- En el campo de la **mecánica estructural**, se considera a la deformación como el principal parámetro a estudiar.

- Entre el estudio del cuerpo rígido y el del mecanismo estructural, se encuentra el extenso campo conocido como **mecánica continua**, donde el movimiento del cuerpo es considerado resultante de un modelo matemático que tiene la desventaja de los casos anteriores.

2.2 Determinación de fuerzas en Sistemas Multicuerpo

Para diseñar las piezas de un Sistema Multicuerpo en cuanto a su resistencia, es necesario determinar las fuerzas y pares de torsión que actúan en los eslabones de forma individual. Cada componente de un sistema completo, por pequeño que sea, deberá analizarse cuidadosamente con respecto a su papel en la transmisión de esfuerzos.

Cuando varios cuerpos se conectan entre sí para formar un grupo o sistema, las fuerzas de acción presentes entre dos cualesquiera de los cuerpos se denominan fuerzas de restricción o fuerzas internas.

Dichas fuerzas obligan o restringen a los cuerpos a comportarse de un modo específico. En cambio, las fuerzas externas que se aplican sobre el sistema de cuerpos se llaman fuerzas aplicadas o fuerzas externas.

Las características que definen a una fuerza son su magnitud, dirección y su punto de aplicación. La dirección de una fuerza incluye el concepto de recta soporte, que es la recta a lo largo de la cual se dirige, así como su sentido. Por ello, una fuerza puede estar dirigida positiva o negativamente a lo largo de una línea de acción [3].

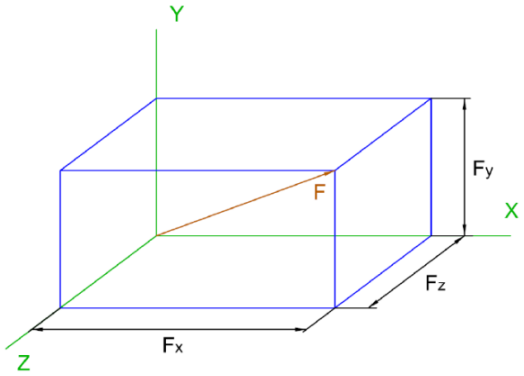

Figura 2.4. Componentes del vector fuerza

Las componentes de un vector fuerza (véase figura 2.4) se escribirán así:

$$\vec{F} = F_x \cdot \vec{i} + F_y \cdot \vec{j} + F_z \cdot \vec{k} \qquad (ec\ 1)$$

Dos fuerzas cualesquiera que actúen en un cuerpo, constituyen un par, en donde el brazo del par, es la distancia perpendicular entre las líneas de acción de las fuerzas aplicadas. El plano del par es aquel que contiene a ambas líneas de acción.

2.2.1 Diagrama de cuerpo libre

Un diagrama de cuerpo libre o diagrama de cuerpo aislado debe mostrar todas las fuerzas externas que actúan sobre el cuerpo. Es fundamental que el diagrama de cuerpo libre esté correcto antes de aplicar la Segunda ley de Newton:

$$\sum \overrightarrow{F_{ext}} = m \cdot \vec{a} \qquad (ec\ 2)$$

En estos diagramas, se escoge un objeto, cuerpo o miembro del sistema mecánico y se aísla, reemplazando las barras, superficies u otros elementos por fuerzas representadas por flechas que indican sus respectivas direcciones. Por supuesto, también debe representarse la fuerza de gravedad y las fuerzas de fricción [5].

Si intervienen varios cuerpos, se hace un diagrama de cada uno de ellos, por separado (véase figura 2.5).

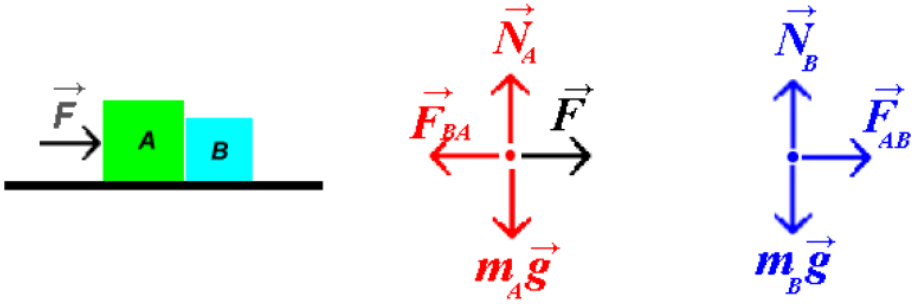

Figura 2.5. Representación de fuerzas en un diagrama de cuerpo libre

El diagrama de cuerpo libre es una herramienta muy utilizada para el análisis de fuerzas en los Sistemas Multicuerpo. Es un esquema o dibujo de un cuerpo aislado de la máquina o mecanismo en el que se representan las fuerzas y los momentos de torsión que actúan en cada pieza. Se deben incluir en el diagrama las magnitudes y las direcciones conocidas, así como cualquier otra información pertinente, tal como las fuerzas externas y/o momentos de torsión externos que actúen sobre el eslabón.

El diagrama obtenido de esta manera se conoce como "libre" ya que se ha separado la parte o porción del cuerpo del resto de los elementos de la máquina y se han reemplazados su efectos por fuerzas y momentos que actúan sobre él.

2.2.2 Análisis de fuerzas en sistemas multicuerpo

Para poder realizar el análisis de fuerzas en un Sistema Multicuerpo completo, generalmente, se debe hacer un diagrama de cuerpo libre de cada eslabón que compone el sistema para indicar las fuerzas que están actuando sobre él. Para determinar las direcciones, sentidos y magnitudes de estas fuerzas, se deben recordar las siguientes leyes de la estática:

- Un cuerpo rígido sobre el que actúan dos fuerzas está en equilibrio estático sólo si las dos fuerzas son colineales y de igual magnitud, pero de sentido opuesto. Si sólo se conocen los puntos de aplicación de las dos fuerzas, como los puntos A y B de la figura 2.6, las direcciones de las dos fuerzas se pueden determinar a partir de la dirección de la línea que une el punto A con el B.

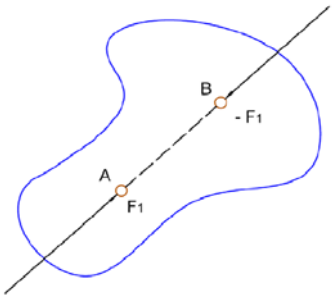

Figura 2.6. Cuerpo en equilibrio estático

- En un cuerpo rígido sobre el que actúan tres fuerzas, éste estará en equilibrio estático, si las líneas de acción son concurrentes en algún punto y la suma de las 3 fuerzas vale cero, tal como se observa en la figura 2.7.

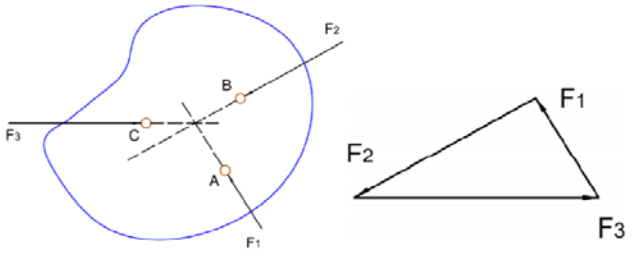

Figura 2.7. Cuerpo rígido con tres fuerzas en equilibrio estático

- Un cuerpo rígido sobre el que actúa un par está en equilibrio estático, sólo si actúa sobre él otro par coplanar, igual en magnitud y en sentido opuesto al primero, tal como se muestra en la figura 2.8.

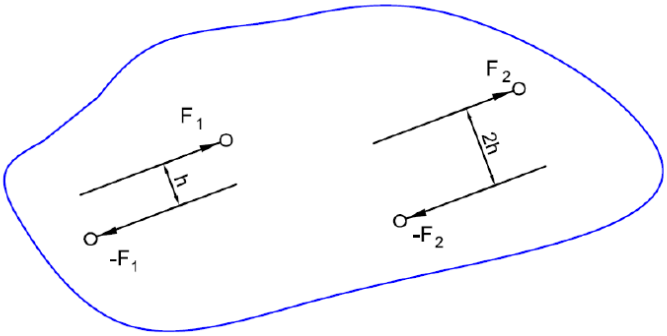

Figura 2.8. Cuerpo rígido en equilibrio estático con dos momentos de torsión

Si más de tres fuerzas actúan sobre un cuerpo en equilibrio estático o si actúan sobre él combinaciones de fuerzas y momentos de torsión, el principio de superposición puede usarse en conjunto con las tres leyes de la estática, es decir el efecto de cada fuerza o momento puede analizarse independientemente y el efecto de todas las fuerzas y momentos de torsión, será la suma vectorial de las resultantes de todos los análisis individuales.

Para el análisis estático de mecanismos compuestos de eslabones rígidos implicará el uso de diagramas de cuerpo libre, así como también, la aplicación de las leyes de la estática.

Cuando se realiza un análisis de fuerzas estáticas, la suma vectorial de las fuerzas en cada eslabón debe ser igual a cero para que permanezca en equilibrio. Lo mismo que se debe cumplir para un análisis dinámico, cuando se emplean tanto fuerzas de inercia como fuerzas externas, las cuales se obtiene a partir de la segunda ley de Newton. Por lo tanto, es conveniente usar el concepto de fuerzas de inercia ya que tanto en los casos estáticos como en los dinámicos se pueden tratar de la misma manera [1].

2.3 Elementos de unión

Como se dijo en la introducción del capítulo 2, los sistemas multicuerpo están unidos mediante elementos de fuerza o pares cinemáticos, los cuales introducen algunas restricciones al movimiento en las uniones de los cuerpos.

2.3.1 Elementos de fuerza

Los elementos de fuerza más importantes son: resortes, amortiguadores y actuadores. A continuación, se describen los distintos elementos de fuerza.

2.3.1.1 Resortes

Los resortes son los elementos de fuerza más utilizados en los sistemas mecánicos. La acción de un resorte da lugar a la aparición de sendas fuerzas en los puntos de anclaje, con la misma línea de acción, el mismo módulo y sentidos opuestos.

La dirección de dichas fuerzas es la que une los puntos de anclaje situados en dos cuerpos distintos del sistema, y el módulo es igual al producto de la constante de rigidez del resorte, *k*, por el alargamiento que el resorte ha experimentado, Δl. Si el muelle se encuentra comprimido, las fuerzas que aparecen trataran de alejar los dos cuerpos, mientras que si está traccionado intentarán acercarlos [2].

Según lo anterior, se puede expresar las fuerzas debidas a la acción del resorte entre dos cuerpos i y j como:

$$\overrightarrow{F_{i_{resorte}}} = k \cdot (l - l_0) \cdot \vec{u} \qquad (ec\ 3)$$

$$\overrightarrow{F_{j_{resorte}}} = -k \cdot (l - l_0) \cdot \vec{u} \qquad (ec\ 4)$$

siendo:

***lo*:** longitud inicial del resorte

***l*:** longitud actual

***k*:** coeficiente de rigidez

***u*:** un vector unitario en la dirección que une ambos puntos de anclaje

En la figura 2.9 se muestra un resorte que forma un sistema multicuerpo.

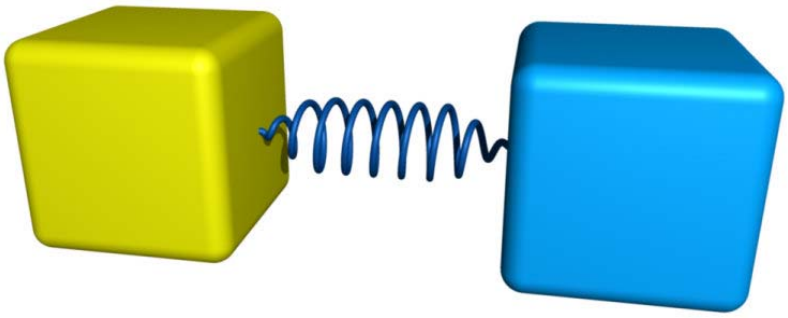

Figura 2.9. Resorte

2.3.1.2 Amortiguadores

Los amortiguadores se comportan de manera similar a los resortes. La diferencia es que los amortiguadores dan lugar a la aparición de fuerzas que se oponen a la velocidad relativa entre sus extremos, y no al desplazamiento relativo entre los mismos [2].

Para los amortiguadores se definen las siguientes expresiones:

$$\overrightarrow{F_{l_{amortiguador}}} = c \cdot \dot{l} \cdot \vec{u} \qquad (ec\ 5)$$

$$\overrightarrow{F_{J_{amortiguador}}} = -c \cdot \dot{l} \cdot \vec{u} \qquad (ec\ 6)$$

donde:

$\dot{l}$: diferencia de velocidad por el amortiguador

c: coeficiente de amortiguamiento

$\vec{u}$: un vector unitario en la dirección que une ambos puntos de anclaje

En la figura 2.10 se muestra un amortiguador dentro de un sistema multicuerpo.

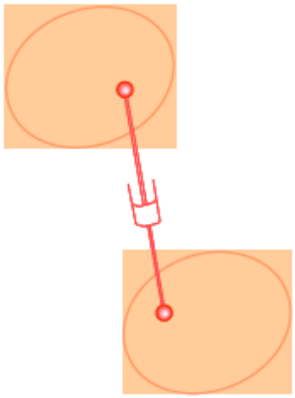

Figura 2.10. Amortiguador

2.3.1.3 Actuadores

El último elemento de fuerza son los actuadores que proporcionan una pareja de fuerzas. Estas fuerzas pueden ser constantes o dependientes del tiempo sobre los cuerpos a los que están anclados.

Como en los elementos de fuerza anteriores, estas fuerzas comparten la misma línea de acción y tienen el mismo módulo pero sentidos opuestos. En la figura 2.11 se representa un actuador de un sistema multicuerpo [6].

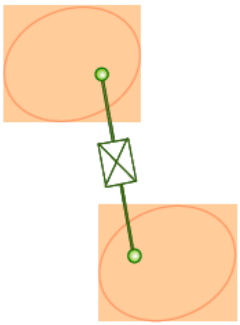

Figura 2.11. Actuador

2.3.2 Pares cinemáticos

Una junta o par cinemático es la conexión que existe entre dos o más eslabones, la cual se encuentra en los nodos de los eslabones y permite algún movimiento o movimiento potencial, entre los eslabones conectados. Las juntas o pares cinemáticos pueden ser clasificados de la siguiente forma [4]:

- Por el número de grados de libertad permitidos en la junta.
- Por el tipo de contacto que existe entre los elementos.
- Por el tipo de cierre de la junta, de fuerza o de forma.
- Por el número de eslabones que están conectados.

A continuación, se describen los distintos tipos de pares cinemáticos.

2.3.2.1 Pares de revolución

En los pares de revolución, un elemento gira respecto al otro, con un único posible movimiento de rotación alrededor de un eje de revolución concreto.

Como se observa en la figura 2.12, el cuerpo 2 únicamente puede girar respecto al cuerpo 1 alrededor de un eje determinado. Si se considera el movimiento de ambos elementos en el plano, este tipo de par tiene cuatro grados de libertad: el giro del elemento dos respecto al uno y los tres propios del elemento uno [2].

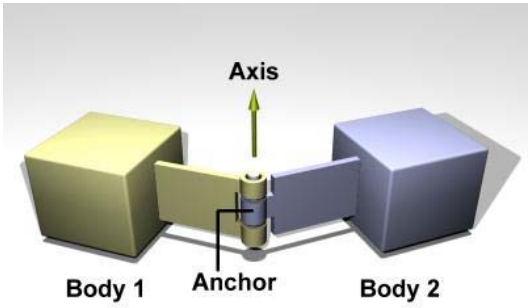

Figura 2.12. Par de revolución

Si ambos elementos estuvieran libres, y por lo tanto no constituyesen un par cinemático, tendrían un total de seis grados de libertad (tres cada uno); por lo que el par de revolución disminuye en dos el número de grados de libertad.

2.3.2.2 Pares esféricos

El par esférico está formado por un sólido esférico que se mueve dentro de otro de la misma forma o viceversa. Su posible movimiento es una rotación alrededor del centro de la esfera (véase figura 2.13).

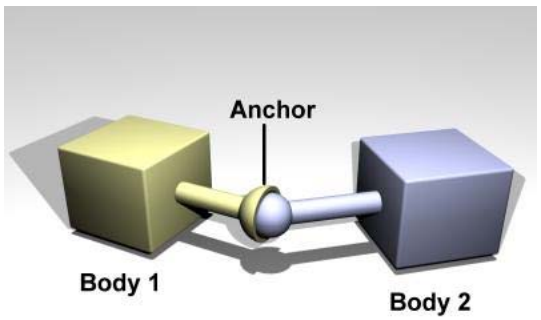

Figura 2.13. Par esférico

Como se aprecia en la figura, el elemento 2 puede girar de manera libre respecto a cualquiera de los tres ejes, sin desplazarse respecto al elemento 1. Con

este tipo de par se dispone de tres posibles rotaciones, una rotación según cada uno de los ejes de coordenadas, tiene tres grados de libertad frente a los seis generales del cuerpo libre [2].

2.3.2.3 Pares prismáticos

Los pares prismáticos o de traslación se componen de un cuerpo prismático que se mueve dentro de otro cuerpo hueco de la misma forma. El movimiento queda restringido a la traslación paralela a las aristas del prisma.

Como se puede ver en la figura 2.14, el elemento 2 únicamente puede trasladarse respecto al 1 en una dirección determinada. Si se considera el movimiento de ambos cuerpos en el plano, el sistema mecánico que forman tiene cuatro grados de libertad: los tres correspondientes al elemento 1, libre, más el grado de libertad asociado al movimiento relativo de deslizamiento del segundo eslabón respecto al primero (el desplazamiento) [2].

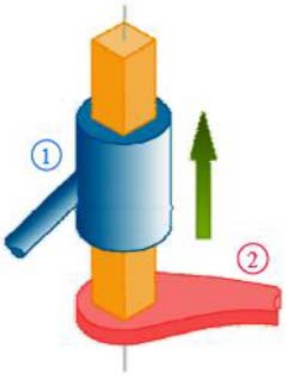

Figura 2.14. Par prismático

2.3.2.4 Pares cilíndricos

Los pares cilíndricos están formados por un sólido con una parte cilíndrica que se mueve dentro de otro sólido con la misma forma. Los posibles movimientos son de rotación alrededor del eje de la unión y una traslación a lo largo de la misma.

En la figura 2.15 se muestra este tipo de par cinemático, se puede ver como el cuerpo 1 puede girar libremente respecto al elemento 2, además también puede desplazarse de manera longitudinal respecto a este.

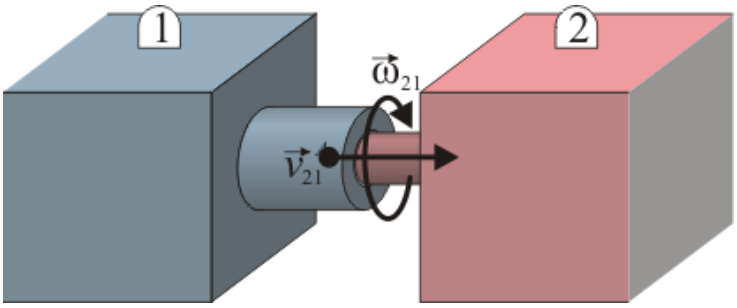

Figura 2.15. Par cilíndrico

El par cilíndrico permite la rotación angular y la traslación de forma independiente de los elementos, por lo tanto, posee dos grados de libertad [2].

2.3.2.5 Pares helicoidales

Los pares helicoidales se componen de un sólido con una parte cilíndrica dotada de un filete helicoidal, que se mueve dentro del agujero correspondiente labrado en el otro sólido. Las superficies de contacto son helicoidales, de manera que permiten entre los dos miembros un movimiento de traslación y uno de rotación relacionados linealmente (véase figura 2.16).

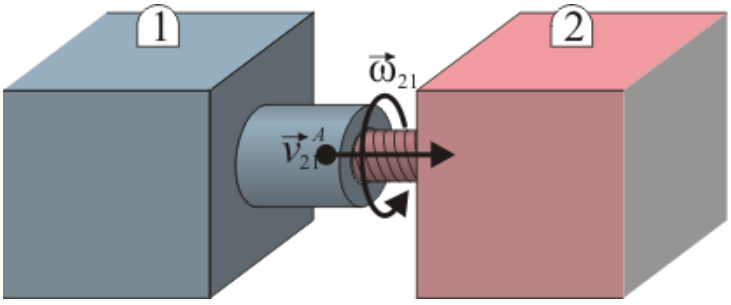

Figura 2.16. Par helicoidal

El miembro que tiene la superficie de contacto exterior, rosca exterior, se denomina tornillo o barra roscada y el que tiene la superficie de contacto interior, rosca interior, se denomina tuerca.

El par helicoidal permite los movimientos relativos de rotación y traslación, aunque posee un solo grado de libertad por estar los dos movimientos relacionados entre sí [2].

2.3.2.6 Pares planos

Los pares cinemáticos planos están constituidos por un sólido de bases planas paralelas, que se mueve entre los dos planos de sus bases. El movimiento relativo es una traslación paralela a los citados planos (véase figura 2.17).

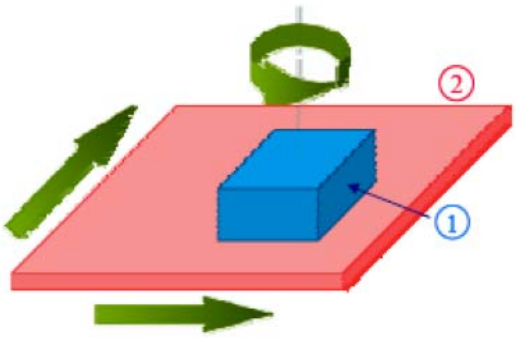

Figura 2.17. Par plano

Las superficies de contacto son planas, de manera que permiten dos traslaciones y una rotación alrededor de una dirección perpendicular al plano de contacto de un miembro respecto al otro, las tres independientes entre ellas. Por lo tanto, deja tres grados de libertad relativos entre los miembros.

2.3.2.7 Pares cinemáticos superiores

En estos pares, el contacto se establece a través de un único punto o de una generatriz recta en superficies regladas. Estos contactos pueden ser con deslizamiento o sin él. El contacto puntual se puede establecer entre:

- Un mismo punto de un miembro y un mismo punto del otro miembro.
- Un mismo punto de un miembro y un punto de una superficie fija al otro miembro.
- Puntos variables de cada uno de los sólidos.

Hay un número infinito de pares cinemáticos superiores, por lo que estos no pueden clasificarse fácilmente.

Los pares cinemáticos superiores más frecuentes son:

- Par guía-corredera.
- Par leva-seguidor.
- Palancas rodantes.
- Engranajes.

En la figura 2.18 se representa un ejemplo de par cinemático superior, se trata de un engranaje tipo piñon-cremallera [4][2].

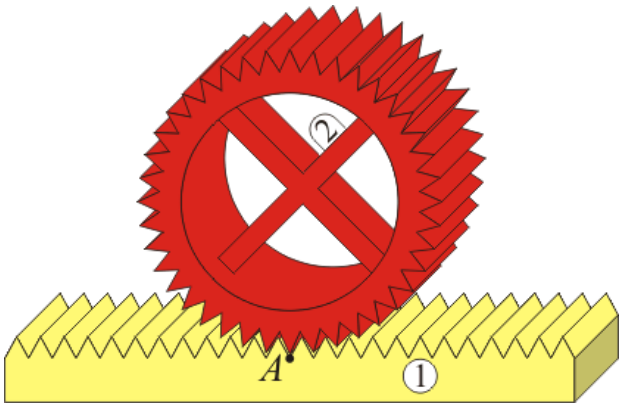

Figura 2.18. Par cinemático superior

2.4 Tipos de coordenadas y sistemas

Cuando se trabaja con sistemas multicuerpo es muy importante tener claros los diferentes tipos de sistemas de coordenadas. La elección del sistema de coordenadas más apropiado para el tipo de mecanismo que se desea analizar influye en la complejidad de la formulación y en el número de ecuaciones para plantear el problema [4][1].

Se definen dos grandes grupos de coordenadas:

- **Coordenadas Independientes:** dan lugar a una modelización con tantas coordenadas como grados de libertad tiene el sistema. Al definir la posición del mecanismo pueden presentar problemas de indeterminación.

- **Coordenadas Dependientes:** en este tipo de grupo se utiliza un número de coordenadas mayor al número de grados de libertad. La diferencia entre las coordenadas y los grados de libertad es la cantidad de restricciones necesarias para definir correctamente el sistema.

Dentro de las Coordenadas Dependientes existen varios grupos de sistemas: coordenadas cartesianas locales, coordenadas cartesianas relativas, coordenadas básicas, coordenadas naturales y coordenadas mixtas.

2.4.1 Coordenadas Cartesianas Locales

Normalmente, para identificar la posición de cada uno de los elementos del sistema multicuerpo respecto al sistema de referencia global XYZ se recurre a la definición de sendos sistemas cartesianos, (ξ, η, ζ) ligados a cada uno de los elementos que componen el sistema multicuerpo, y que evolucionan solidariamente con el miembro al que se han fijado (véase figura 2.19).

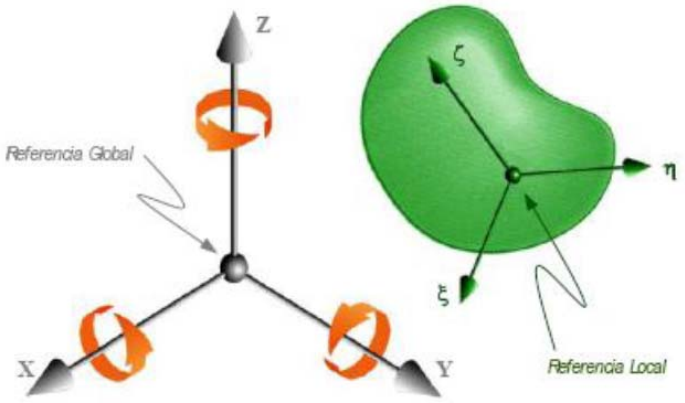

Figura 2.19. Sistema de coordenadas Local

Para especificar la posición de cada cuerpo con estas coordenadas, basta con indicar las coordenadas cartesianas globales del punto en el que se encuentra el origen de la referencia local que evoluciona solidariamente con él.

En cuanto a su orientación, en los sistemas planos es suficiente con definir un ángulo que indique la orientación del sistema local respecto al global, mientras que en el espacio se necesitarán tres coordenadas angulares que definan unívocamente la orientación de los ejes locales respecto a los globales [2] [4].

2.4.2 Coordenadas Cartesianas Relativas

Las coordenadas cartesianas relativas son aquellas que expresan las distancias X e Y respecto al último punto capturado. Cuando se trabaja con coordenadas relativas, en lugar de definir la posición y orientación de cada elemento respecto al sistema global de referencia, lo que se hace es referirlas al miembro contiguo, mediante un ángulo, una distancia, o bien mediante combinaciones de ambos. Así, las distancias se asocian con los grados de libertad que permiten el desplazamiento relativo entre ambos cuerpos, mientras que los ángulos se aplican en los casos en que existe posibilidad de giro [2][4].

Normalmente, para posicionar cada uno de los elementos con las coordenadas cartesianas relativas se necesitan menos parámetros que en el caso de emplear coordenadas locales. Concretamente, tan sólo se necesita una variable por

cada uno de los grados de libertad permitidos en el movimiento relativo entre el cuerpo que se pretende localizar y el cuerpo respecto al cual se están refiriendo las coordenadas [5].

Una vez conocidas la posición y orientación globales de uno de los miembros del sistema multicuerpo, así como las coordenadas relativas correspondientes a todos los demás, es posible determinar la posición exacta de todos ellos en el sistema global.

Este tipo de coordenadas resulta especialmente apropiado para trabajar con sistemas multicuerpo abiertos; sin embargo, para sistemas cerrados conducen a una formulación bastante más compleja que la que se obtendría en el caso de definir la configuración del sistema mediante coordenadas locales.

2.4.3 Coordenadas Básicas

Las coordenadas básicas se pueden considerar como una alternativa a los dos tipos de coordenadas anteriores. Su filosofía de utilización consiste en definir la configuración del sistema tan sólo mediante las coordenadas globales de ciertos puntos significativos pertenecientes a cada uno de sus miembros, a los que se llamaran puntos primarios.

Utilizando estas coordenadas, la posición y orientación de un cuerpo queda definida por al menos dos puntos, necesitándose a lo sumo cuatro puntos en el caso más general. Así, un cuerpo unidimensional (como podría ser una barra de pequeño diámetro), se representaría por dos puntos primarios; un cuerpo bidimensional (como por ejemplo una placa delgada), podría definirse mediante tres puntos; y uno tridimensional por cuatro puntos (véase figura 2.20). Y conociendo las coordenadas globales de todos los puntos primarios de un sólido, se podrían calcular la de cualquier otro punto [2] [4].

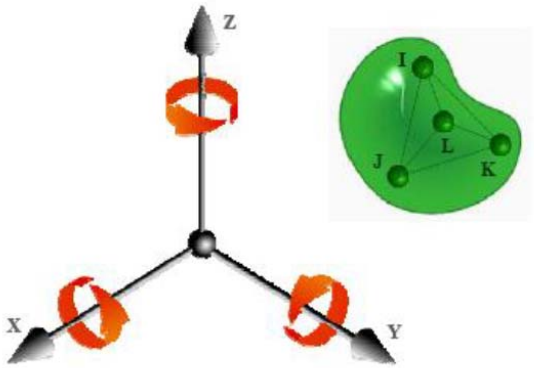

Figura 2.20. Sistema de coordenadas básicas

Con este tipo de coordenadas se puede definir la orientación de un cuerpo sin necesidad de recurrir a la utilización de coordenadas angulares, que tienen un tratamiento complicado al trabajar en tres dimensiones. Esto se traduce en una extraordinaria sencillez en la formulación de las relaciones de dependencia entre las distintas coordenadas, que han de tenerse en consideración sea cual sea el sistema de coordenadas elegido, y que se deben a la existencia de restricciones geométricas que limitan el movimiento relativo entre miembros contiguos de un sistema multicuerpo, ligados entre sí por juntas cinemáticas.

Otra de sus ventajas es que resultan igualmente apropiadas para sistemas abiertos o cerrados, cosa que no ocurría con las coordenadas relativas.

Las coordenadas básicas pueden ser consideradas como una evolución de las coordenadas de punto de referencia, en donde los puntos de referencia se desplazan hasta las juntas de unión de los pares cinemáticos. De esta forma, los puntos de referencia son siempre compartidos por dos elementos.

2.4.4 Coordenadas Naturales

Las coordenadas naturales son una evolución de las coordenadas básicas, se las podría considerar como el resultado de sustituir algunas parejas de puntos primarios correspondientes a un mismo cuerpo, por uno tan sólo de ellos, más un vector que va desde uno de estos puntos hasta el otro, con la diferencia de que en la práctica se emplean vectores unitarios (véase figura 2.21).

Con estas coordenadas, cada uno de los miembros de un sistema multicuerpo quedaría definido mediante un conjunto de puntos y vectores, cuyas coordenadas y componentes, referidas al sistema global de referencia, definirían de forma unívoca su posición y orientación en el espacio [2] [4].

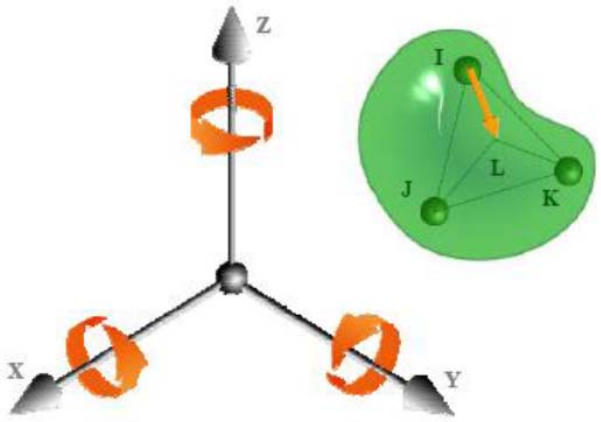

Figura 2.21. Coordenadas naturales

Las coordenadas naturales presentan las mismas cualidades que las coordenadas básicas, con la ventaja añadida de que la utilización de vectores resulta especialmente apropiada para modelizar las relaciones de dependencia que existen entre las distintas coordenadas.

2.4.5 Coordenadas Mixtas

Las coordenadas mixtas son una combinación de los tipos de coordenadas anteriores, y se emplean en algunos casos en los que interesa conocer determinado tipo de desplazamientos relativos entre dos miembros del sistema multicuerpo.

Una modelación se ha llevado a cabo en coordenadas mixtas si además de coordenadas naturales se han utilizado algunas coordenadas relativas, ángulos y distancias. El empleo de coordenadas relativas junto con las coordenadas naturales resulta de gran interés en muchos casos. Por ejemplo, son de enorme utilidad a la hora de definir pares como el par de engranaje, dado que este par impone una cierta proporcionalidad en los valores de los giros, o si se desean considerar hilos inextensibles, que implican una ecuación de suma de distancias.

En otros casos, el mero hecho de disponer del valor de un cierto ángulo o distancia directamente como resultado del análisis, sin necesidad de postproceso, ya justifica la utilización de coordenadas mixtas [2] [4].

2.5 Fundamentos teóricos

La base teórica de los métodos dinámicos actualmente utilizados en la simulación de sistemas multicuerpo se apoya en la mecánica clásica, concretamente en el planteamiento de las ecuaciones diferenciales del movimiento de sistemas de varios sólidos rígidos con restricciones.

En 1687 Newton sentó los cimientos de la mecánica al describir de forma completa la mecánica de un punto material sometido a fuerzas centrales. Sin embargo, no se encuentra en su obra una descripción del movimiento de los cuerpos extensos, ya sean o no rígidos.

En ese momento comienza a desarrollarse la mecánica del sólido, y fue Euler en 1776 el que por primera vez precisó matemáticamente los conceptos de masa puntual y aceleración y desarrolló el concepto de “sólido rígido”, lo que le permitió completar las ecuaciones de Newton con las ecuaciones que gobiernan el movimiento de rotación. Ambas ecuaciones constituyen las llamadas “ecuaciones de Newton-Euler”, que son las ecuaciones diferenciales que gobiernan el movimiento 3-D de un único sólido rígido.

Al trabajo de Euler se sumó el de Bernouilli, D`Alembert y Lagrange, sus trabajos se convertirían en los principios fundamentales de la mecánica, desplazando la mecánica newtoniana pura de Euler y comenzando la línea de desarrollo que culminaría con la elaboración de la llamada "mecánica clásica".

En 1743, D`Alembert estudio por primera vez la dinámica de varios sólidos rígidos con restricciones (“sistema multicuerpo” en la actualidad), distinguiendo entre fuerzas “aplicadas” y fuerzas “de reacción”, e intuyó el principio de los trabajos virtuales.

Lagrange profundizó en la mecánica analítica, la cual no se fundamenta en los principios de Newton o Euler, sino en los cuatro principios de conservación de la dinámica conocidos en su tiempo: conservación de las fuerzas vivas, conservación del movimiento del centro de gravedad, conservación del momento de la cantidad de movimiento y principio de la mínima acción.

En 1788 Lagrange, quien aplicando un principio variacional a la suma de las energías cinética y potencial, llegó a las ecuaciones diferenciales del movimiento prácticamente en la forma en que hoy se utilizan [7].

Las ecuaciones de Lagrange constituyen una formidable síntesis de la mecánica precedente junto a la nueva formulación de las ecuaciones del movimiento de los cuerpos y soluciones originales a problemas realmente difíciles.

Las ecuaciones de Lagrange, también conocidas como Ecuaciones de Euler-Lagrange, o simplemente de Euler, permiten contar con un sistema analítico para llegar a las ecuaciones que describen el comportamiento físico de las partículas, pero no se trata, de ningún modo, de una nueva teoría independiente de la teoría Newtoniana.

Las ecuaciones de Euler-Lagrange para la partícula j-ésima se definen como:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = F_j \qquad (ec\ 7)$$

donde:

- q_j son las coordenadas generalizadas de posición, puede ser una distancia, un ángulo, etc.
- $\dot{q}_j$ son las velocidades generalizadas, son las derivadas temporales de las posiciones.
- F_j son las fuerzas generalizadas que actúan sobre la partícula j.
- La función **L** $(q_j , \dot{q}_j)$ se llama Lagrangiano y se define como **L=T-V** donde **T** es la energía cinética total del sistema y **V** es la energía potencial del sistema:

$$T = \sum_{i=1}^{n} T_i \qquad (ec\ 8)$$

$$V = \sum_{i=1}^{n} V_i \qquad (ec\ 9)$$

La mecánica lagrangiana es una reformulación de la mecánica clásica, en ella la trayectoria de un objeto es obtenida encontrando la trayectoria que minimiza la acción, que es la integral del Lagrangiano en el tiempo, siendo este la energía cinética del objeto menos la energía potencial del mismo. La formulación lagrangiana simplifica considerablemente muchos problemas físicos.

Para poder aplicar estas ecuaciones de una manera eficiente en campos como la ingeniería (principalmente en el campo de la automoción) se hace necesario el uso de ordenadores digitales, apareciendo las formulaciones que dan lugar a programas informáticos basados en coordenadas relativas.

Sin embargo, poco a poco se iban a abrir paso las formulaciones "globales", mucho más sencillas y versátiles para abordar el estudio de las topologías arbitrarias que surgían en la industria del automóvil, aplicando el uso de coordenadas de puntos de referencia, utilizando ángulos de Euler para determinar la orientación angular.

Tras la solución del problema dinámico inverso, los esfuerzos se concentraron en el problema dinámico directo o problema de la simulación dinámica, que consiste en calcular las aceleraciones en función de las fuerzas, para unas posiciones y velocidades dadas.

Para la resolución del problema dinámico directo se recurrió a métodos recursivos, que no tenían la necesidad de resolver ningún sistema de ecuaciones lineales y podía resolver el problema con un número de operaciones aritméticas proporcional al número de grados de libertad [7].

Capítulo 3

Modelado

3.1 Modelado orientado a objetos

El modelado orientado a objetos es un potente e intuitivo paradigma para la construcción de modelos, que proporciona al modelador opciones para ocultar la complejidad (encapsulación), para permitir la reutilización (herencia y agregación) y crear modelos independientes de fácil mantenimiento.

El desarrollo modular permite a un sistema ser modelado de abajo hacia arriba. Los componentes básicos de la librería se pueden combinar para crear componentes cada vez más complejos mediante la combinación de dos métodos:

- Extensión por herencia de los componentes existentes.
- Agregación de los componentes existentes.

El modelado orientado a objetos (MOO) ha heredado las características propias de los lenguajes de programación orientados a objetos (POO), en los que se basa [3] [6].

3.1.1 Características del modelado orientado a objetos

3.1.1.1 Encapsulación

Uno de los objetivos del modelado orientado a objetos es el encapsulamiento de la complejidad. Los lenguajes de programación orientada a objetos (POO) hacen la vida más fácil al maquetista con su gran capacidad de abstracción y la encapsulación de datos y comportamiento.

Sus elementos principales son las librerías y componentes. Las librerías encapsulan todos los elementos que intervienen en una determinada disciplina, y se exponen para el uso de otras librerías. El componente es un elemento fundamental para el modelado para expresar un comportamiento dinámico asociado a ciertos datos.

Los componentes están conectados por sus puertos. Al definir el comportamiento de los puertos, el sistema es capaz de insertar de forma automática varias ecuaciones de conexión, por lo que el usuario no tiene por qué estar preocupado por ellas. Para agregar un nuevo componente, sólo hay que preocuparse por su comportamiento interno y no de las ecuaciones de conexión.

3.1.1.2 Herencia

La herencia es lo que da a estos lenguajes (POO) su gran poder para el intercambio de interfaces y comportamiento. En todos los campos de la simulación, cuando se han desarrollado muchos componentes se pone de manifiesto que comparten gran parte de la conducta. Este lenguaje puede reunir datos comunes y ecuaciones en los componentes padre, para ser heredados por los componentes hijo.

Esta característica permite la creación de librerías basadas en componentes padre con un orden lineal en vez del complejo orden geométrico. Un nuevo componente sobre la base de otro padre, incluirá todos estos datos y comportamientos.

Estos lenguajes pueden ofrecer también herencia múltiple, con la cual un componente puede heredar los datos y el comportamiento de varios componentes que previamente han sido diseñados y probados. Esta capacidad permite la creación de nuevos componentes reutilizando partes de otros que ya se han creado.

3.1.1.3 Agregación

En los lenguajes POO, todos los elementos de modelado son los componentes, un componente puede heredar de otros y puede contener copias internas de otros. Este concepto refuerza la capacidad de reutilización porque permite crear componentes compuestos sobre la base de otros que ya se han desarrollado.

Este paradigma se aplica de forma iterativa y no tiene límite. La complejidad de un componente final puede estar oculta por la agregación de probadas y eficaces instancias internas de otros componentes o clases.

3.1.2 Tipos de modelos

Existen dos principios básicos y bastante diferentes para la construcción de modelos:

- **Modelado físico:** consiste en descomponer el sistema en subsistemas con propiedades conocidas. Para sistemas técnicos se utilizan las leyes naturales que los describen. Las características fundamentales de los modelos obtenidos de esta forma son: la posibilidad de incorporación de los mecanismos básicos de funcionamiento, la dificultad de su obtención y la necesidad de la experiencia y el conocimiento de los sistemas bajo estudio.

- **Identificación:** el otro principio básico consiste en usar las observaciones de los sistemas para conseguir fijar las propiedades del modelo a las del sistema. En general se usa como complemento del método anterior. Entre sus características generales se encuentran las siguientes: generan normalmente modelos lineales, no se hacen hipótesis previas, no se tienen en cuenta los mecanismos internos del sistema y se basan solamente en datos experimentales de entrada-salida.

Los diferentes tipos de modelos matemáticos se pueden clasificar en cuatro categorías: modelos de tiempo continuo, modelos de tiempo discreto, modelos cualitativos y modelos de sucesos o eventos discretos [6].

3.1.2.1 Modelos de tiempo continuo

Los modelos de tiempo continuo se caracterizan porque en intervalos finitos de tiempo, las variables de estado cambian sus valores infinitas veces. De esta forma, los modelos de este tipo están representados por conjuntos de ecuaciones diferenciales. De entre los modelos de tiempo continuo se pueden distinguir dos categorías distintas [6]:

- Modelos de parámetros concentrados. Están descritos por ecuaciones diferenciales ordinarias (EDO).
- Modelos de parámetros distribuidos. Están descritos por ecuaciones en derivadas parciales (EDP).

En la Figura 3.1 se muestra cómo una variable cambia a lo largo del tiempo en un modelo de tiempo continuo.

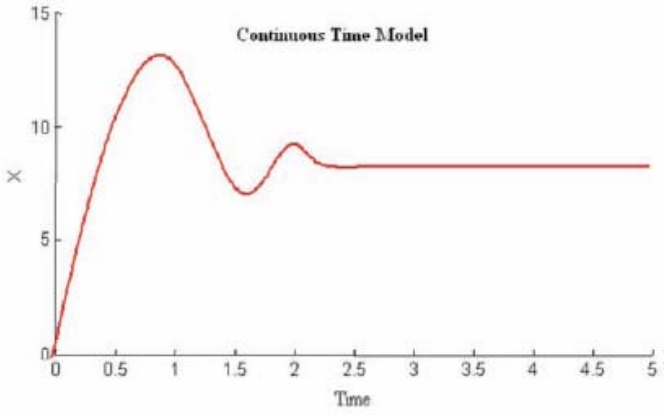

Figura 3.1. Respuesta de un modelo en tiempo continuo

3.1.2.2 Modelos de tiempo discreto

En los modelos de tiempo discreto el eje temporal se discretiza. Se representan comúnmente por ecuaciones en diferencias (EDif), al menos cuando la discretización está espaciada de forma equidistante.

Los modelos de tiempo discreto se utilizan frecuentemente en ingeniería, sobre todo en sistemas controlados por ordenador. Los modelos de tiempo discreto pueden ser también versiones discretizadas de sistemas continuos. Este es un hecho bastante común. De hecho, cuando se utiliza un ordenador para simular un modelo continuo, realmente se tiene que discretizar el eje de los tiempos para evitar el problema de los infinitos cambios [6].

En la Figura 3.2 se muestra la trayectoria exhibida por un modelo de tiempo discreto.

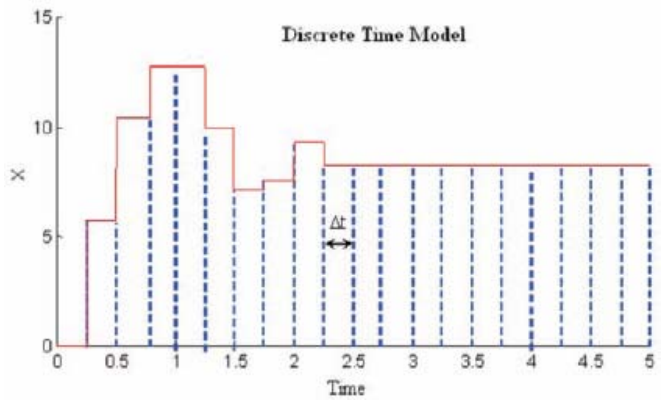

Figura 3.2. Respuesta de un modelo en tiempo discreto

3.1.2.3 Modelos cualitativos

Los modelos cualitativos son, en realidad, modelos discretos, aunque con la discretización del eje temporal no necesariamente equidistante. En la Figura 3.3 se muestra una trayectoria seguida por un modelo de esta categoría.

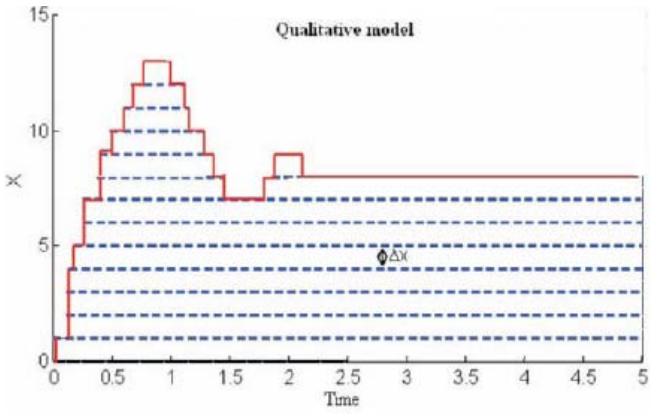

Figura 3.3. Respuesta de un modelo cualitativo

3.1.2.4 Modelos de sucesos o eventos discretos

En los modelos de sucesos o eventos discretos, tanto el eje temporal como el eje de estado son continuos, pero difieren de los modelos de tiempo continuo en que sólo puede ocurrir un número finito de cambios. En la figura 3.4 se puede observar una trayectoria típica de este tipo de modelos.

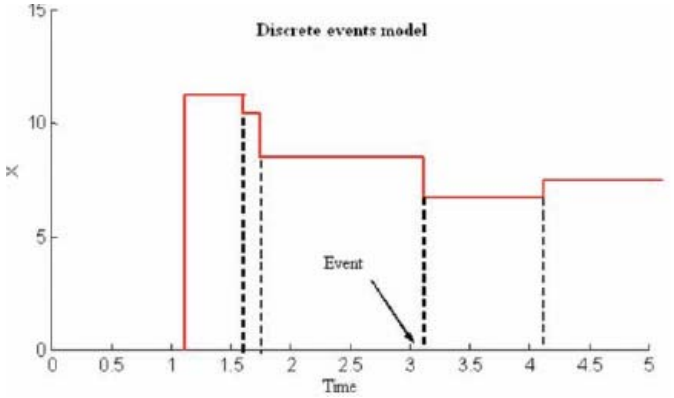

Figura 3.4. Respuesta de un modelo de eventos discretos

Dentro de este tipo se incluyen los modelos de colas de cajas en grandes almacenes, determinados procesos de fabricación, la llegada de los coches a un peaje, etc. Este tipo de modelos requiere de grandes dosis de matemática estadística [6].

3.1.3 Modelado causal y acausal

La causalidad es la relación entre un evento (la causa) y un segundo evento (el efecto), donde se entiende el segundo evento, como consecuencia de la primera.

En el modelado hay dos tipos básicos de conexiones:

- Las conexiones acausales, en las que el flujo de datos en la conexión no se especifica explícitamente.

- Las conexiones causales, también llamado conexiones de señal, donde se especifica la dirección del flujo de datos utilizando la entrada de prefijos en la declaración de al menos uno de los puertos en la conexión.

Las conexiones causales suelen surgir cuando hay una señal que fluye en una dirección determinada. Un sistema **causal** es un sistema con salida y estados internos que sólo depende de los valores de entrada actuales y anteriores.

El modelado **acausal** es un estilo de modelado declarativo, es decir, modelos basados en ecuaciones en lugar de sentencias de asignación. Las ecuaciones no especifican cuáles son las variables de entrada y ni de salida. En la figura 3.5 se muestra un ejemplo de sistema acausal.

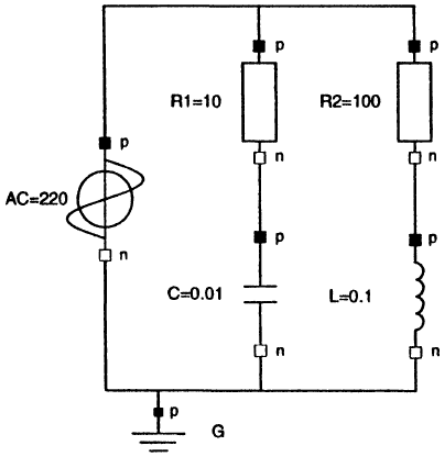

Figura 3.5. Diagrama de conexión de un modelo de circuito acausal

3.2 Introducción al modelado con EcosimPro

3.2.1 Introducción al programa

El programa que se ha utilizado para la modelización ha sido EcosimPro. Se trata de un potente software capaz de modelar cualquier tipo de sistema dinámico representado por ecuaciones diferenciales algebraicas u ordinarias. En él están integrados entornos visuales que proporcionan una herramienta intuitiva para facilitar la simulación.

Las herramientas proporcionan una orientación a objetos no casual enfocada para la creación de librerías de componentes reutilizables. Se basan en métodos numéricos y simbólicos capaces de procesar complejos sistemas de ecuaciones diferenciales algebraicas.

EcosimPro se encarga internamente de extraer las ecuaciones del modelo final, transformarlas simbólicamente, detectar problemas de exceso de variables, resolver automáticamente lazos algebraicos lineales y no lineales y reducir el índice (por medio de derivación simbólica) cuando se encuentre problemas sobredeterminados [8].

Este programa dispone de poderosos resolvedores de ecuaciones lineales, no lineales y de ecuaciones algebraico-diferenciales. Todo esto permanece oculto al modelador, con lo que se puede concentrar en el sistema físico a modelar, dejando a EcosimPro preocuparse de los aspectos complejos de manejo y optimización del modelo matemático.

También la capacidad de crear nuevos componentes gráficamente usando una herramienta CAD. Para cada componente se puede crear un icono y después reusarlos para crearse por medio de arrastrar, soltar y conectar nuevos componentes

EcosimPro ha sido usado en entornos industriales muy exigentes como la ESA, NASA, Snecma, DASA, Alenia, etc. Tiene múltiples aplicaciones en campos como: control, turborreactores, celdas de combustible, balances térmicos, quimica, tratamiento de aguas y residuos, mecánica, economía, redes termales, robótica, etc. También puede ser usada como puro resolvedor de ecuaciones algebraico-diferenciales.

Las librerías de componentes en EcosimPro son totalmente reusables por otros usuarios, pues no incluyen causalidad computacional en las ecuaciones (es decir el formato con el que se introducen las ecuaciones es intrascendente, pues el programa despejara automáticamente la variable que más le convenga en cada caso). Los nuevos usuarios pueden reusar librerías existentes o crear sus propias librerías.

El modelado dinámico orientado a objetos es relativamente nuevo, ya que su uso no comenzó a generalizarse hasta los años 90. Su uso está todavía limitado a una pequeña minoría, la mayoría de las personas del sector utilizan códigos secuenciales como aproximaciones con bloques, ecuaciones, etc.

El modelado orientado a objetos de sistemas físicos sólo abarca los conceptos y prácticas que han existido en la programación convencional desde finales de los años 80. Las ventajas de la utilización de este tipo de modelado para resolver problemas de complejidad media y alta han sido más que demostradas.

El lenguaje de programación orientada a objetos de EcosimPro, conocido por EL, es uno de los pioneros que tiene que hacer frente a esta nueva forma de modelado de sistemas físicos [8].

Las ventajas de este tipo de metodología son:

- El modelador encapsula los datos y el comportamiento en los componentes individuales (se minimizan los datos globales).
- Un componente oculta la complejidad de hacer público solo una parte del componente.
- La interfaz pública comprende los parámetros, datos y puertos, mientras que las variables locales, eventos discretos y ecuaciones siguen siendo privados.
- La complejidad crece de forma lineal en vez de forma geométrica.
- Reutilización de componentes probados.
- La herencia simplifica el modelado mediante el intercambio de datos comunes y ecuaciones. Con menos código se consigue mayor productividad.
- Un componente puede contener ecuaciones que se pueden insertar en el momento de la simulación o no, dependiendo de ciertos parámetros que se pasan al componente.
- El formato de ecuaciones es declarativo. Los algoritmos transforman simbólicamente las ecuaciones de modo que se ajusten de la mejor manera posible para ser resueltas numéricamente.

Los componentes probados se reutilizan constantemente a través de herencia simple o múltiple. Es más fácil hacer cambios en los modelos porque todo se divide en partes.

Los principales objetivos a la hora de modelar un sistema dinámico son:

- **Velocidad:** cumplir los requisitos de cada aplicación.
- **Corrección:** resultados comparados con los reales.
- **Robustez:** los modelos no tienen problemas numéricos.

- **Extensibles:** fácil de extender en el futuro.
- **Modificables:** sencillos de modificar.
- **Legibles:** listas de modelo sencillas y fáciles de seguir.
- **Modular:** componentes y puertos divididos en librerías.
- **Reutilización:** poder reutilizar componentes mediante herencia.
- **Compatibles:** tratar de estandarizar los elementos como unidades de las variables.
- **Integridad:** especificar las restricciones en la integridad del modelo para prevenir la propagación de errores.

Para mejorar la extensibilidad se debe simplificar el diseño para que sea más fácil adaptarse a los cambios. También es importante la descentralización, haciendo que los módulos tengan la mayor autonomía posible y que un simple cambio afecte sólo a un módulo o a un número pequeño de ellos. Si no, el cambio podría dar lugar a una reacción en cadena.

Estos factores determinan la calidad en la programación de un sistema dinámico.

Los componentes de EcosimPro se pueden construir desde cero o formarlos a partir de componentes básicos predefinidos. Cada componente contiene una descripción matemática de los componentes del mundo real que representa. Los componentes se conectan mediante puertos de conexión para crear un nuevo componente.

Los experimentos definen las condiciones iniciales y las condiciones de contorno del modelo matemático, y las soluciones deseadas. Estos experimentos pueden ser triviales o muy complejos utilizando lenguaje secuencial.

El proceso de construcción de componentes en EcosimPro tiene los siguientes pasos:

- Crear las librerías
- Codificar componentes, puertos, etc.
- Compilar en una librería
- Generar una partición
- Crear un experimento
- Simular

3.2.2 Características del programa

Las principales características que proporciona EcosimPro son [8]:

- Simulación dinámica multidisciplinar.
- Potente manipulación simbólica y numérica de ecuaciones.
- Modelado orientado a objetos.
- Comprobación de la integridad de simulación en cualquier momento por medio de afirmaciones y de rangos de variables.
- Acceso directo a las funciones externas en FORTRAN, C y C + +.
- Optimizado para trabajar con miles de ecuaciones.

En la Figura 3.6 se observa un ejemplo de un experimento creado en EcosimPro.

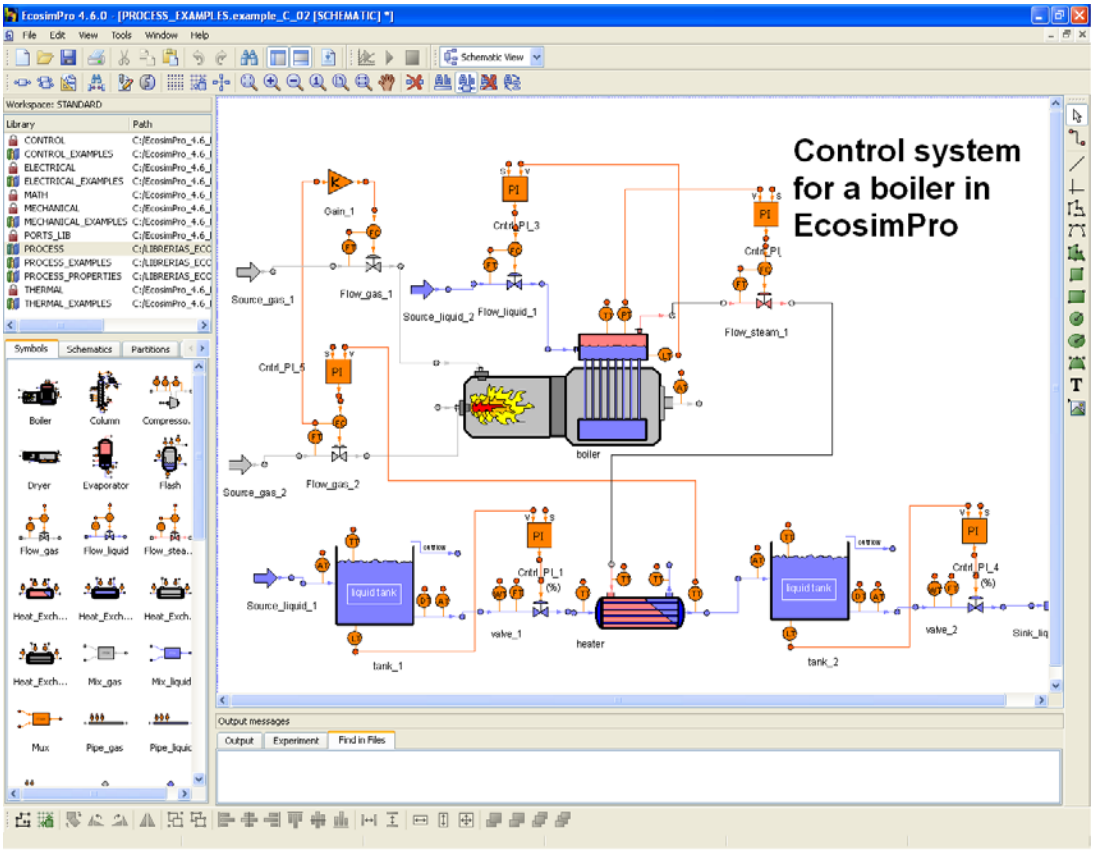

Figura 3.6. Experimento en EcosimPro

3.2.3 Lenguaje de modelado

EcosimPro tiene un lenguaje de modelado llamado EL (EcosimPro Language). Este lenguaje se utiliza para modelar sistemas dinámicos.

Las principales ventajas de este lenguaje son:

- Modelado no-causal.
- Sintaxis simple para su modelo de componentes.
- Lenguaje de simulación orientado a objetos, con herencia simple y múltiple entre componentes.
- Se pueden modelar todo tipo de eventos discretos del sistema.

- Puertos de conexión para simplificar la definición de componentes y la topología.

- Lenguaje experimental para crear múltiples casos de simulación para el mismo modelo en el mismo experimento.

- La programación declarativa para las ecuaciones de entrada. No importa cuál sea el formato de la ecuación de entrada, EcosimPro lo transformará simbólicamente cuando sea necesario.

El lenguaje de simulación (EL) tiene una gran variedad de características del lenguaje para la expresión del modelo. Este lenguaje proporciona instrucciones secuenciales similares a Fortran, C, etc; así como estados continuos y discretos para modelar el comportamiento físico.

El programa compilador comprueba la corrección léxica, sintáctica y semántica de cualquier nuevo componente. Si el compilador detecta un error, informa del problema con el número de línea y ofrece una explicación para el error.

EL proporciona capacidades para la definición de los componentes, puertos de conexión, funciones, tipos de enumeración, etc. El último modelo de simulación se traduce en código C + + para su uso en otros módulos de software y ejecutar las simulaciones [8].

3.2.4 Tipos de datos básicos

Para el modelado, todas las variables deben declararse antes de su uso para que el compilador sepa de qué tipo son. Todos los identificadores tienen un tipo asociado, que determina las operaciones que pueden realizarse en ellos y cómo deben ser interpretadas dichas operaciones [8].

En EcosimPro/EL se pueden encontrar cuatro categorías de datos:

- Tipos fundamentales como Real o Boolean, se utilizan para todo tipo de operaciones lógicas y aritméticas.

- Tipos de enumeración como Enum.

- Tipos de derivados como arrays y Set_of.

- Tipos de datos complejos como Component, Port y Class. Estos son los elementos con un alto nivel de abstracción que contienen las variables, ecuaciones, etc.

3.2.5 Librerías

Las librerías son un mecanismo utilizado para organizar el diseño de la información. EcosimPro proporciona un mecanismo de librerías muy bueno para el manejo de distintos tipos de componentes y fácil de expresarlo.

Con éste programa todo va a ser incorporado en una librería. En general, las librerías están clasificadas por disciplina como Eléctricas, Control, Mecánicas, etc.

El programa viene con un conjunto de librerías estándar y otras que se pueden adquirir por separado. En la figura 3.7 se puede ver la configuración típica de las librerías que se encuentran en EcosimPro.

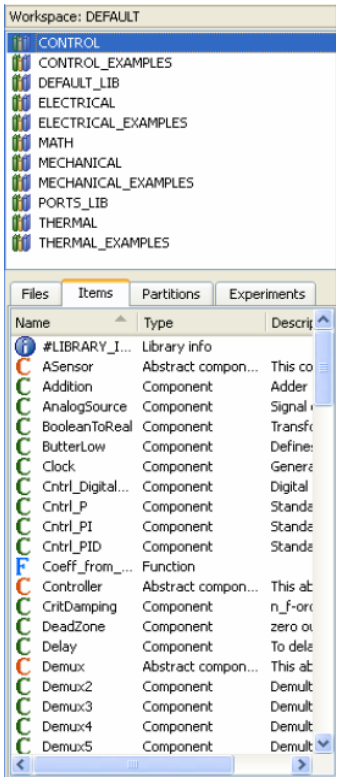

Figura 3.7. Librerías EcosimPro

Una librería de EL es un conjunto de elementos que pueden ser componentes, tipos de puertos, variables globales, clases y funciones que están relacionados en el mismo área [8].

EcosimPro permite al usuario manipular las librerías creando nuevas o desactivando otras.

3.2.6 Declaraciones

En EL se pueden utilizar tres tipos de declaraciones, secuencial, continua o discreta. Los lenguajes clásicos como Fortran y C++ sólo permiten la declaración secuencial, otros lenguajes orientados a objetos únicamente permiten las declaraciones discretas.

Normalmente EcosimPro utiliza el modelado continuo, pero también se puede utilizar para simular componentes discretos o la ejecución de códigos secuenciales [8].

3.2.6.1 Declaraciones secuenciales

Las declaraciones secuenciales se utilizan para la inicialización, funciones y cuerpos de eventos discretos que requieren una estricta ejecución de las órdenes. Se ejecutan de manera progresiva y permiten controlar el flujo del programa.

Las declaraciones secuenciales permitidas son las típicas que se encuentran en otros lenguajes de programación como C++, ADA, etc. Se componen de asignaciones, bucles iterativos y sentencias condicionales.

3.2.6.2 Declaraciones continuas

Las declaraciones continuas se utilizan para expresar conjuntos de ecuaciones diferenciales algebraicas, donde el orden en el que se escriben no es importante y serán ordenados más tarde por los algoritmos internos. Estos forman el núcleo donde los modelos físicos continuos se definen.

Las declaraciones continuas son las que definen los modelos matemáticos en la parte continua, pero estos modelos se verán afectados por los sucesos en la parte discreta. Cuando el modelo continuo está integrando, cualquier acontecimiento que se indique es observado.

3.2.6.3 Declaraciones discretas

Las declaraciones discretas se utilizan para expresar acontecimientos. Estos acontecimientos están controlados por las declaraciones condicionales que indican cuando se produce un evento.

EcosimPro permite que los eventos se definan en la parte continua, estos detienen los modelos continuos para que puedan ser tratados.

La declaración más importante es WHEN, que declara un suceso a la espera de una condición que se convierta en realidad, y en este punto el código asociado es ejecutado. También se encuentran otros tipos de declaraciones discretas como ASSERT, EXPAND, etc.

3.2.7 Puertos

Un puerto de conexión conecta los componentes por los valores que cada uno necesita para comunicarse con su entorno, y por las restricciones y ecuaciones que se aplican a las conexiones. Mediante la definición de nuevos puertos se puede modelar cualquier tipo de intercambio de valores entre componentes.

El mecanismo básico para el modelado de un sistema de manera modular es conectar los componentes entre sí utilizando sus puertos. Las conexiones crean un componente compuesto. Desde fuera el componente compuesto se puede utilizar para conectarse a sus puertos externos [8].

Cada puerto de un componente debe tener un modo, puede ser IN u OUT. Estos modos tienen tres propósitos:

- Sirven como referencia para el signo de las variables de flujo. Si es positivo el puerto IN implica flujo hacia el interior. Si es positivo el puerto OUT indica flujo hacia el exterior.

- Como criterio de aplicación de ecuaciones en las conexiones.

- Se utilizan para comprobar si hay violaciones de las restricciones de conexión especificadas en el puerto.

El nombre del puerto se declara con un identificador. Un puerto también puede contener información especial acerca de:

- Herencia: Indica que un cuerpo hereda de uno o más puertos.

- Parámetros de construcción: no se pueden cambiar durante la simulación.

- Descripción opcional del puerto en un formato de texto.

3.2.7.1 Ecuaciones de conexión de los puertos

Se necesitan las ecuaciones de conexión de los puertos en los siguientes casos:

- Cuando un puerto está conectado a más de un puerto.

- Cuando un puerto está conectado a otro puerto en el mismo nodo, IN-IN, OUT-OUT.

En ambos casos, los puertos no pueden considerarse equivalentes, por lo tanto, es necesario añadir las ecuaciones de conexión que relacionan las variables de los puertos implicados. Las ecuaciones de conexión se generan para cada variable del puerto.

Existen tres tipos de comportamiento básicos:

- EQUAL: las variables con este comportamiento mantienen el mismo valor para todos los puertos de conexión.
- SUM: en una conexión la suma de los flujos entrantes es igual al flujo de salida.
- Sin indicación: no se genera ninguna ecuación de conexión para la variable.

3.2.7.2 Restricciones de conexión de los puertos

En algunos modelos, son útiles para especificar algunas restricciones sobre la conexión entre puertos.

Hay dos grupos de restricciones:

- Límites en el número de puertos que pueden ser conectados entre sí.
- Límites en los modos de los puertos conectados entre sí.

Las restricciones se incluyen por dos razones, que el desarrollador las incluya o la falta de información en la definición de las variables.

3.2.8 Componentes

Los componentes son los elementos más importantes de EcosimPro/EL, son el instrumento básico para la representación de los modelos simples y complejos. Un componente es una manera natural de describir el comportamiento tanto de los modelos continuos como discretos.

Los componentes representan los objetos físicos que actúan independientemente, pero que están conectados a otros. Pueden ser tan simples como una ecuación continua o el manejo de un evento discreto [8].

3.2.8.1 Componentes abstractos

Los componentes en EL pueden ser abstractos o concretos. Los componentes abstractos describen una conducta que por sí sola no representa a ningún elemento físico, y sólo puede ser utilizado como un componente base para otros componentes.

Un componente abstracto puede contener las mismas ecuaciones que tienen los componentes normales. La única diferencia con los normales es que no se puede llegar a crear un modelo final. Son muy útiles en el modelado de cualquier campo ya que ahorran muchas líneas de código al reunir la interfaz y las ecuaciones que comparten varios componentes.

3.2.8.2 Categorías de las variables

Se pueden clasificar las variables fundamentales de los componentes en tres categorías:

- En el bloque PARAMETER, los parámetros de construcción se utilizan sólo en una instancia del componente (público).
- En el bloque DATA, hay variables de los componentes que normalmente se conocen (público).
- En el bloque DECLS, hay variables locales para el componente (privado).

3.2.8.3 Bloque discreto

La información discreta consiste en eventos que deben ser detectados durante la simulación. Cuando se detecta, el sistema ejecuta una secuencia de acciones asociadas que pueden a su vez activar nuevos eventos en una reacción en cadena. Cuando no hay más eventos el sistema sigue con la parte continua, si existe, o se detiene.

En este bloque, sólo se pueden utilizar las declaraciones discretas. Los eventos discretos se evalúan en cada instante de tiempo de los avances de la parte continua.

3.2.8.4 Bloque continuo

El bloque continuo contiene la parte continua del componente, aquí es donde se definen las ecuaciones diferenciales algebraicas. El problema matemático se manda al solucionador numérico de EcosimPro.

Las variables que se calculan en las ecuaciones se definen como continuas, porque su valor sigue el valor dado por la ecuación matemática, en lugar de cambiar de forma esporádica.

3.2.9 Funciones

Las funciones en EcosimPro son similares a las de otros lenguajes de programación como Fortran o C. Una función es una pieza del código donde se llevan a cabo operaciones y opcionalmente, devuelve un valor. Estas se utilizan para una amplia gama de problemas como:

- Evaluar una forma numérica y devolver un valor.
- Reutilizar las funciones probadas.
- Encapsular el código de uso frecuente.

Las funciones en EL deben ser definidas antes de ser utilizadas, y las funciones externas escritas en otros lenguajes han de ser pre-declaradas. Una función se compone de una declaración del interfaz, la definición de variables locales y del cuerpo de la función [8].

3.2.9.1 Funciones con forma de onda

EcosimPro proporciona un conjunto de funciones que normalmente se utilizan para generar las entradas a las variables de la simulación. Estas son muy útiles pero difíciles de programar, ya que son discontinuas. En ocasiones, la discontinuidad no es muy importante o no tiene mucha influencia en el sistema de ecuaciones, y puede ser ignorada. En dichas funciones, las discontinuidades son detectadas automáticamente por el programa y se manejan correctamente para continuar con la simulación.

Estas son las funciones que vienen predefinidas en EcosimPro:

- ***Square***: esta función genera una forma de onda cuadrada (véase figura 3.8).

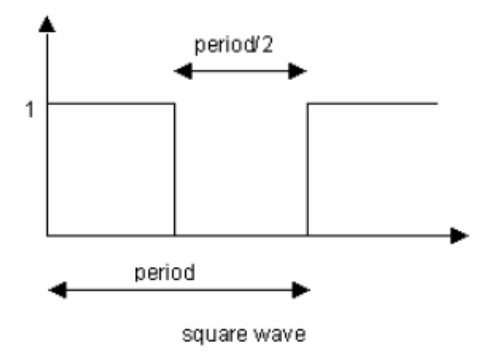

Figura 3.8. Función *Square*

- ***Step****:* esta función genera una forma de onda de paso (véase figura 3.9).

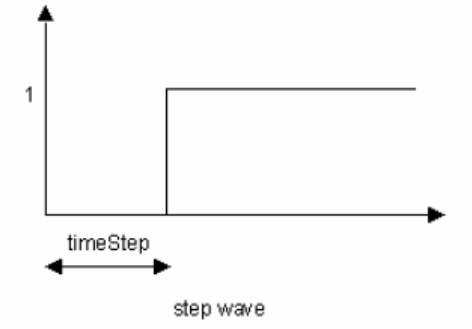

Figura 3.9. Función *Step*

- ***Pulse****:* esta función genera una forma de onda de pulso (véase figura 3.10).

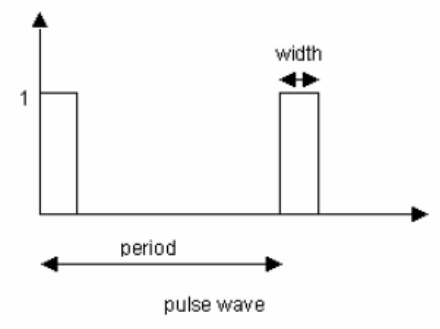

Figura 3.10. Función *Pulse*

- ***Ramp**:* esta función genera una forma de onda rampa (véase figura 3.11).

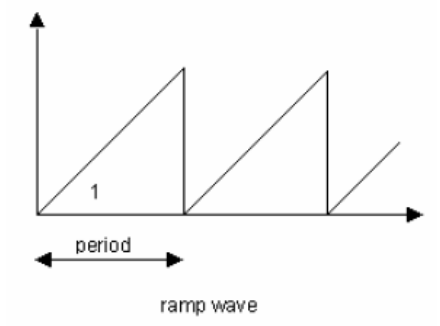

Figura 3.11. Función *Ramp*

3.2.10 Particiones

Las particiones en EcosimPro son un paso intermedio entre el componente y el experimento. Una partición es un modelo matemático del componente, antes de simular un componente se debe definir una partición y luego crear un experimento. El programa siempre ofrece, por defecto, una partición para cada componente [8].

El programa también ofrece la opción de editar una partición y observar cada una de las condiciones que se han impuesto. Dos particiones serán iguales si todas las condiciones que tienen son idénticas. EcosimPro puede hacer particiones diferentes en función de los siguientes factores:

- Eliminar los derivados del modelo, esto se puede hacer cuando se quiere eliminar todos los aspectos dinámicos del modelo matemático.

- Tener en cuenta las propiedades de las variables, el modelador puede proporcionar propiedades a las variables.

- Cambiar DATA por una variable desconocida, esto se suele utilizar para llevar a cabo los estudios de diseño y dar mayor flexibilidad al programa.

- Seleccionar las condiciones de contorno, cuando el sistema contiene más variables que ecuaciones.

- Óptimo desgarro de bucles algebraicos, el programa ayuda a resolver los lazos algebraicos no lineales de ecuaciones, el modelador debe elegir qué variable debe elegirse como algebraica.

- Resolver problemas de alto grado, seleccionando las variables que no quieren que el solucionador las calcule.

3.2.11 Experimentos

Una vez que el modelo ha sido desarrollado, se puede llevar a cabo experimentos con él. EcosimPro/EL tiene un lenguaje para experimentos que es muy parecido al lenguaje utilizado para funciones y componentes. Su propósito es permitirnos inicializar los datos para el cálculo de los estados estacionarios, o para integrar el modelo en un plazo de tiempo a través del uso de instrucciones secuenciales [8].

En los experimentos se pueden incluir los siguientes bloques:

- DECLS: declaraciones de las variables locales para el experimento.

- OBJECTS: declaración de los objetos externos definidos con clases.

- INIT: valores iniciales asignados a las variables dinámicas.
- BOUNDS: expresiones matemáticas asignadas a las variables de contorno, establece los valores de todos los límites del modelo.
- BODY: instrucciones secuenciales que definen el experimento.

3.2.11.1 Funciones en los experimentos

Las funciones en los experimentos se pueden reutilizar y definirse en las librerías o en el propio experimento. Entre ambas funciones se encuentran algunas diferencias:

- Las funciones del experimento no serán visibles en la librería.
- En las funciones del experimento, se puede acceder directamente a las variables de partición y pueden integrar el modelo.

3.2.12 Comparación entre Modelica y EcosimPro

Modelica y EcosimPro/EL son lenguajes orientados a objetos y acausales basados en ecuaciones para el modelado de sistemas físicos continuos y de eventos discretos. El primero fue desarrollado por una organización sin ánimo de lucro con sede en Suecia. La creación de dicho lenguaje por parte de esta asociación se debe a la necesidad de definir un lenguaje de modelado y simulación estándar, fruto del acuerdo entre un grupo de especialistas de estas disciplinas. Por otra parte, EL es un lenguaje de modelado y simulación de sistemas físicos desarrollado por *EA International* para su herramienta de modelado EcosimPro.

Ambos lenguajes son fundamentalmente declarativos, y contienen todos los elementos necesarios para modelar y simular sistemas dinámicos continuos, discretos e híbridos de manera no causal [3] [8].

3.2.12.1 Tipos y propiedades de los datos

En la tabla 4.1 se describen los tipos de variables fundamentales en ambos lenguajes.

Tabla 4.1. Tipos de variables fundamentales

Tipo	Modelica	EcosimPro/EL
Booleano	Boolean	BOOLEAN
Entero	Integer	INTEGER
Real	Real	REAL
Cadena	String	STRING
Tabla 1D	NO	TABLE_1D
Tabla 2D	NO	TABLE_2D
Tabla 3D	NO	TABLE_3D
Punteros a funciones	NO	FUNC PTR

EL contempla los tipos Tabla y punteros a funciones como tipos base del lenguaje. Modelica no los incorpora como tales, aunque proporciona mecanismos para implementarlos como tipos compuestos.

Propiedades de los datos. Variabilidad

La variabilidad de los distintos tipos de variables está presente en ambos lenguajes. La única diferencia existente entre ambos lenguajes es a nivel sintáctico. Ambos permiten definir variables con distintas variabilidades:

- **Constantes:** Una vez declaradas e inicializadas nunca cambian su valor.
- **Parámetros:** Una vez declaradas e inicializadas no cambian su valor durante una simulación, aunque lo pueden hacer de una simulación a otra.

- **Variables discretas:** Cambian su valor en el transcurso de la simulación solamente en los instantes en los que se producen eventos discretos. En los intervalos de tiempo entre eventos mantienen su valor.

- **Variables continuas:** Cambian su valor en cualquier instante de la simulación.

3.2.12.2 Clases de componentes

Las clases o componentes son el elemento fundamental en los lenguajes de modelado orientado a objetos. Constituyen la herramienta básica para representar el comportamiento, desde sistemas simples e indivisibles, hasta sistemas muy complejos formados por otros subsistemas.

En ambos lenguajes las clases se definen mediante un conjunto de secciones de código. Cada sección tiene un objetivo determinado en cada uno de los lenguajes. En la tabla 4.2 se enumeran las secciones de las clases en ambos lenguajes.

Tabla 4.2. Secciones de los lenguajes

Tipo	Modelica	EcosimPro/EL
Booleano	Boolean	BOOLEAN
Entero	Integer	INTEGER
Real	Real	REAL
Cadena	String	STRING
Tabla 1D	NO	TABLE_1D
Tabla 2D	NO	TABLE_2D
Tabla 3D	NO	TABLE_3D
Punteros a funciones	NO	FUNC PTR

No existe una correspondencia única entre secciones de ambos lenguajes.

Respecto a la inicialización de variables, en Modelica se puede hacer mediante dos vías: asignación de atributos ‘start’ y ‘fixed’ de las variables a inicializar, o definiendo ecuaciones para el instante inicial en la sección ‘initial equation’. En ambos casos las sentencias utilizadas son de carácter declarativo por

lo que la herramienta realizará manipulaciones algebraicas necesarias para resolver el problema de valor inicial planteado en la simulación. Sin embargo, en EL, las inicializaciones se realizan en el bloque INIT, ejecutándose secuencialmente las instrucciones de inicialización contenidas en este bloque [8].

3.2.12.3 Librerías

Las librerías son conceptualmente similares en ambos lenguajes. Su objetivo es el de almacenar modelos o clases ya desarrolladas para posterior utilización. Las diferencias se pueden encontrar en los detalles de implementación.

Una diferencia clara entre EL y Modelica es la inclusión de propiedades de representación gráfica de los componentes dentro de ellos mismos y en las librerías en los que se almacenan. Modelica permite a un componente asociar un icono gráfico dentro de la misma definición de éste mediante directivas 'annotation'. De esta forma, un usuario podría programar la información gráfica de su componente además de modelar su comportamiento [3].

EcosimPro no tiene un capturador de esquemas integrado en la herramienta. En su lugar, se puede utilizar una herramienta externa, SmartSketch, que proporciona al usuario las posibilidades gráficas que EcosimPro no ofrece.

3.2.12.4 Otras diferencias

La **partición** es un concepto solamente implementado en EcosimPro que permite dar una gran flexibilidad al diseño de experimentos y simulaciones. Es una característica implementada en la herramienta y no en el lenguaje. Estas son las posibilidades que ofrecen las particiones [8]:

- Elección de condiciones de contorno.
- Selección de estados en problemas de índice superior.
- Cambiar variabilidad de parámetros.

EcosimPro/EL ofrece explícitamente la posibilidad de definir **experimentos** sobre los sistemas a simular mediante bloques de código secuencial que permite: declarar variables, inicializar variables algebraicas en los modelos, definir condiciones de contorno, cálculo de estacionarios, selección de métodos de integración, selección de tolerancias, generación de informes, invocación de funciones externas, etc.

3.2.12.5 Conclusiones

En general, ambos lenguajes están perfectamente dotados de construcciones que permiten abordar la metodología de modelado orientado a objeto de sistemas continuos, discretos e híbridos. Se puede concluir diciendo:

- EcosimPro/EL aumentaría sus posibilidades de modelado si los arrays pudiesen contener componentes, además de tipos simples y derivados en EL.
- EL aumentaría sus posibilidades de Modelado con la introducción de clases paramétricas.
- El interfaz para simular experimentos de modelos con EcosimPro mediante lenguajes de programación (C++) es una capacidad muy flexible no ofrecida por Modelica.
- La utilización de herramientas gráficas complementarias por parte de EcosimPro/EL, como SmartSketch, hace más incómodo la labor de modelado que si dicha funcionalidad estuviera incorporada en la misma herramienta. Debido a esto, modelar gráficamente en Modelica es mucho más cómodo que en EcosimPro.

Capítulo 4

Creación de una Librería Multicuerpo

En este proyecto, se pretende crear una librería con EcosimPro/EL que permita empezar a trabajar en la programación multicuerpo. En este programa no existía ninguna librería que permitiese programar de tal manera.

Para crear una librería de programación multicuerpo se ha tomado como referencia otro lenguaje de programación orientada a objetos y no-causal llamado Modelica. Para Modelica si que se han desarrollado varias librerías que permiten la programación multicuerpo, este lenguaje ha sido imprescindible para entender este tipo de programación.

Al no disponer de nada desarrollado en EcosimPro, ha sido difícil encontrar la forma de que, tanto el programa como su lenguaje, permitiesen la programación multicuerpo. Además, en el transcurso de la programación multicuerpo con EcosimPro se ha presentado un fuerte contratiempo dado que dicho programa no contempla el tratamiento matricial. Y este tipo de herramienta es básica para la creación de una librería multicuerpo.

La única forma se solventarlo ha sido realizando paso a paso todas las operaciones matriciales del código, con el consiguiente aumento de las líneas de código y la disminución de su eficiencia.

En EcosimPro se ha creado una librería llamada *MULTIBODY* donde se han creado los distintos puertos y componentes necesarios para llevar a cabo la iniciación a la programación multicuerpo. Dentro de esta librería hay tres ficheros, uno que contiene los puertos llamado *PORTS*, otro que contiene los componentes esenciales llamado *COMPONENTS* y el último fichero, que contiene los ejemplos para simular, llamado *Examples*.

En la figura 4.1 se observa la librería *MULTIBODY* con sus diferentes ficheros.

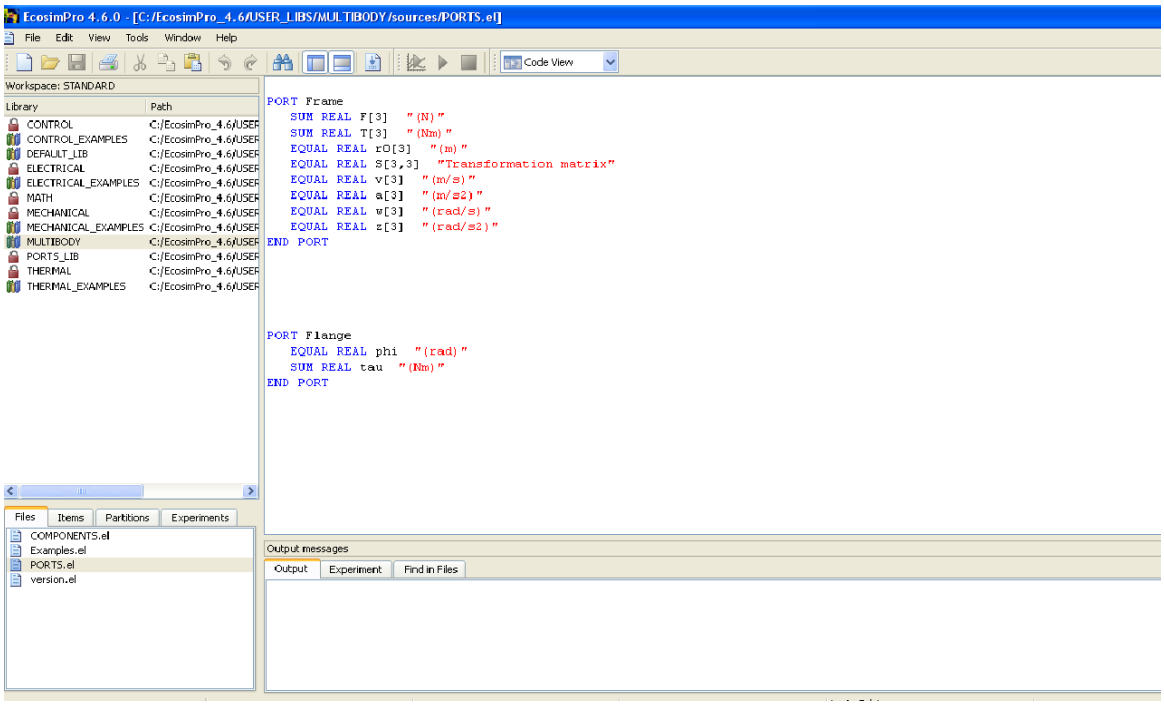

Figura 4.1. Librería *Multibody* en EcosimPro

4.1 Librería de Puertos

En la programación multicuerpo todos los elementos mecánicos y no mecánicos están conectados entre sí mediante puertos. Se trabaja con dos tipos de puertos: **frame** y **flange**.

El puerto **frame** es un sistema de coordenadas en el corte-plano del punto de conexión. Las variables del corte-plano se definen en relación con el sistema de referencia correspondiente [9].

En los puertos **frame** se utilizan las siguientes variables:

- **S:** matriz de rotación que describe el Frame con respecto al Frame inercial.
- **r0:** vector desde el origen del Frame inercial al origen de Frame_A, resuelto en Frame inercial (Los demás vectores se resuelven respecto a Frame_A).
- **v:** velocidad absoluta (traslacional) de Frame_A.
- **w:** velocidad angular absoluta de Frame_A.
- **a:** aceleración absoluta (traslacional) de Frame_A.
- **z:** aceleración angular absoluta de Frame_A.

También se utilizan dos variables de flujo:

- **F:** resultante de corte-fuerza que actúa en el origen de Frame_A.
- **T:** resultante de corte-par con respecto al origen de Frame_A.

A continuación, se escribe el código del puerto **frame**:

```
PORT  Frame
        SUM  REAL  F[3]       "(N)"
        SUM  REAL  T[3]       "(Nm)"
        EQUAL  REAL  r0[3]    "(m)"
        EQUAL  REAL  S[3,3]   "Transformation matrix"
        EQUAL  REAL  v[3]     "(m/s)"
        EQUAL  REAL  a[3]     "(m/s2)"
        EQUAL  REAL  w[3]     "(rad/s)"
        EQUAL  REAL  z[3]     "(rad/s2)"
END  PORT
```

El otro puerto con el que se trabaja se llama **flange,** es un puerto para sistemas de rotación mecánica de una dimensión.

En los puertos **flange** se utilizan las siguientes variables:

- **phi:** ángulo de rotación absoluta del puerto.
- **tau:** resultante de corte-par con respecto al puerto.

A continuación se escribe el código del puerto **flange**:

```
PORT  Flange
        EQUAL  REAL  phi    "(rad)"
        SUM  REAL  tau       "(Nm)"
END PORT
```

4.2 Librería de Componentes

En este apartado, se describen brevemente los componentes desarrollados para la librería multicuerpo de EcosimPro.

En primer lugar, se encuentra el componente **Damper**, es un amortiguador que absorbe energía del sistema. En este programa, se utiliza para disminuir las oscilaciones no deseadas de un movimiento periódico.

El componente **Damper** dispone de dos puertos flange, uno de entrada y otro igual de salida. En este componente "d" es la constante de amortiguación. En la figura 4.2 se muestra este componente.

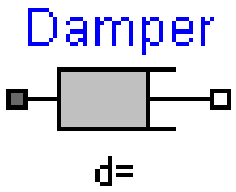

Figura 4.2. Componente Damper

A continuación, se muestra el código desarrollado del componente **Damper**:

```
COMPONENT  Damper
        PORTS
                IN  Flange  flange_A
                OUT  Flange  flange_B
        DATA
                REAL  d = 1        "(N.m.s/rad)"
        DECLS
                REAL  tau          "(Nm)"
                REAL  phi_rel      "(rad)"
                REAL  w_rel        "(rad/s)"
        CONTINUOUS
                phi_rel  =  flange_B.phi  -  flange_A.phi
                flange_B.tau  = tau
                flange_A.tau  =  - tau
                w_rel  =  phi_rel'
                tau  =  d*w_rel
END COMPONENT
```

El siguiente componente se denomina **InertialSystem**, es el sistema de referencia inercial respecto al cual se resuelven los vectores principales de los puertos. El **InertialSystem** siempre debe encontrarse en los sistemas multicuerpo [9].

En la figura 4.3 se observa la representación gráfica del sistema inercial.

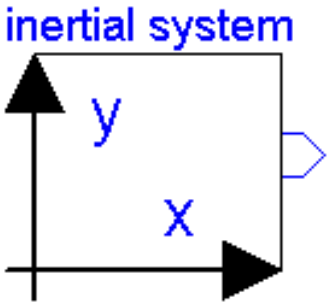

Figura 4.3. Componente InertialSystem

A continuación, se muestra el código desarrollado del componente **InertialSystem**:

```
COMPONENT  InertialSystem
        PORTS
                OUT  Frame  frame_B
        CONTINUOUS
                frame_B.S[1,1]  =  1
                frame_B.S[1,2]  =  0
                frame_B.S[1,3]  =  0
                frame_B.S[2,1]  =  0
                frame_B.S[2,2]  =  1
                frame_B.S[2,3]  =  0
                frame_B.S[3,1]  =  0
                frame_B.S[3,2]  =  0
                frame_B.S[3,3]  =  1
                frame_B.a[1]  =  0
                frame_B.a[2]  =  -9.8
                frame_B.a[3]  =  0
                EXPAND_BLOCK  ( i  IN  1,3 )
                        frame_B.r0[i]  =  0
                        frame_B.v[i]  =  0
                        frame_B.w[i]  =  0
                        frame_B.z[i]  =  0
                END  EXPAND_BLOCK
END  COMPONENT
```

El siguiente componente se denomina **TwoTreeFrames**, es un componente que no puede utilizarse sólo, sino que otros componentes heredan sus propiedades para simplificar la programación.

Este componente dispone de dos puertos frame, uno de entrada (A) y otro de salida (B). Se utiliza para igualar las variables entre los puertos que se unen de distintos componentes, y para continuar el flujo de ciertas variables, por ejemplo el par, en los componentes que se unen [9].

En la figura 4.4 se observa la representación gráfica de **TwoTreeFrames**.

Figura 4.4. Componente TwoTreeFrames

A continuación, se muestra el código desarrollado del componente **TwoTreeFrames**:

```
ABSTRACT  COMPONENT  TwoTreeFrames
        PORTS
                OUT  Frame  frame_B
                IN  Frame  frame_A
        DECLS
                REAL  Fa[3]    "(N)"
                REAL  Ta[3]    "(Nm)"
                REAL  Fb[3]    "(N)"
                REAL  Tb[3]    "(Nm)"
                REAL  Sa[3,3]    "Transformation matrix"
                REAL  Sb[3,3]    "Transformation matrix"
                REAL  r0a[3]    "(m)"
                REAL  va[3]    "(m/s)"
                REAL  wa[3]    "(rad/s)"
                REAL  aa[3]    "(m/s2)"
                REAL  za[3]    "(rad/s2)"
                REAL  r0b[3]    "(m)"
                REAL  vb[3]    "(m/s)"
                REAL  wb[3]    "(rad/s)"
                REAL  ab[3]    "(m/s2)"
                REAL  zb[3]    "(rad/s2)"
        CONTINUOUS
                frame_A.S  =  Sa
```

```
                frame_A.r0 = r0a
                frame_A.v = va
                frame_A.w = wa
                frame_A.a = aa
                frame_A.z = za
                frame_A.F = Fa
                frame_A.T = Ta
                frame_B.S = Sb
                frame_B.r0 = r0b
                frame_B.v = vb
                frame_B.w = wb
                frame_B.a = ab
                frame_B.z = zb
                frame_B.F = Fb
                EXPAND ( i IN 1,3) frame_B.T[i] = (-1)*Tb[i]
END COMPONENT
```

El siguiente componente se denomina **BoxBody**. Es un cuerpo rígido con forma de caja, las propiedades de la masa del cuerpo se calculan de los datos de la caja (véase figura 4.5).Éste componente hereda las propiedades del componente **TwoTreeFrames.** Los vectores del componente **BoxBody** se definen con respecto a Frame_A.

Estas son las variables que se declaran en **BoxBody** [9]:

- **r:** vector de posición desde el origen de Frame_A al origen de Frame_B.
- **r0:** vector de posición desde Frame_A al punto medio del plano izquierdo de la caja.
- **LengthDir:** Vector unitario en la dirección de la longitud.
- **WidthDir:** Vector unitario en la dirección de la anchura.
- **L:** Longitud de la caja.
- **W:** Anchura de la caja.
- **H:** Altura de la caja.
- **Wi:** Ancho de la superficie de la caja interior.
- **Hi:** Altura de la superficie de la caja interior.
- **rho:** Densidad del material.

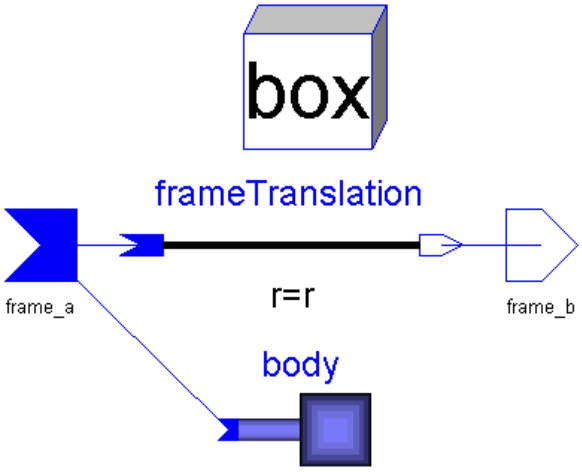

Figura 4.5. Componente BoxBody

A continuación, se muestra el código desarrollado del componente **BoxBody**:

```
COMPONENT  BoxBody  IS_A TwoTreeFrames
        DATA
                REAL  r[3]  =  {0.8,0,0}     "(m)"
                REAL  r0[3]  =  {0,0,0}      "(m)"
                REAL  L  =  0.8    "(m)"
                REAL  LengthDir[3]  =  {0.8,0,0}    "(m)"
                REAL  WidthDir[3]  =  {0,1,0}       "(m)"
                REAL  W  =  0.1      "(m)"
                REAL  H  =  0.1      "(m)"
                REAL  Wi  =  0       "(m)"
                REAL  Hi  =  0       "(m)"
                REAL  rho  =  7.7    "(g/cm3)"
                REAL  zero =  0
        DECLS
                REAL  mo      "(kg)"
                REAL  mi      "(kg)"
                REAL  I[3,3]    "(kg.m2)"
                REAL  m       "(kg)"
                REAL  rCM[3]     "(m)"
                REAL  vaux[3]     "(m/s)"
                REAL  Iw[3]
                REAL  prod1[3]
                REAL  prod2[3]
                REAL  prod3[3]
                REAL  pro1[3]
                REAL  pro2[3]
                REAL  pro3[3]
```

```
        REAL  pro4[3]
CONTINUOUS
        pro1[1]  =  - za[3]*rCM[2] + za[2]*rCM[3]
        pro1[2]  =  za[3]*rCM[1] - za[1]*rCM[3]
        pro1[3]  =  - za[2]*rCM[1] + za[1]*rCM[2]
        pro2[1]  =  - wa[3]*rCM[2] + wa[2]*rCM[3]
        pro2[2]  =  wa[3]*rCM[1] - wa[1]*rCM[3]
        pro2[3]  =  - wa[2]*rCM[1] + wa[1]*rCM[2]
        Iw[1]  =  I[1,1]*wa[1] + I[1,2]*wa[2] + I[1,3]*wa[3]
        Iw[2]  =  I[2,1]*wa[1] + I[2,2]*wa[2] + I[2,3]*wa[3]
        Iw[3]  =  I[3,1]*wa[1] + I[3,2]*wa[2] + I[3,3]*wa[3]
        pro3[1]  =  - wa[3]*Iw[2] + wa[2]*Iw[3]
        pro3[2]  =  wa[3]*Iw[1] - wa[1]*Iw[3]
        pro3[3]  =  - wa[2]*Iw[1] + wa[1]*Iw[2]
        pro4[1]  =  - rCM[3]*Fa[2] + rCM[2]*Fa[3]
        pro4[2]  =  rCM[3]*Fa[1] - rCM[1]*Fa[3]
        pro4[3]  =  - rCM[2]*Fa[1] + rCM[1]*Fa[2]
        prod1[1]  =  - za[3]*r[2] + za[2]*r[3]
        prod1[2]  =  za[3]*r[1] - za[1]*r[3]
        prod1[3]  =  - za[2]*r[1] + za[1]*r[2]
        prod2[1]  =  - wa[3]*vaux[2] + wa[2]*vaux[3]
        prod2[2]  =  wa[3]*vaux[1] - wa[1]*vaux[3]
        prod2[3]  =  - wa[2]*vaux[1] + wa[1]*vaux[2]
        prod3[1]  =  - r[3]*Fa[2] + r[2]*Fa[3]
        prod3[2]  =  r[3]*Fa[1] - r[1]*Fa[3]
        prod3[3]  =  - r[2]*Fa[1] + r[1]*Fa[2]
        vaux[1]  =  - wa[3]*r[2] + wa[2]*r[3]
        vaux[2]  =  wa[3]*r[1] - wa[1]*r[3]
        vaux[3]  =  - wa[2]*r[1] + wa[1]*r[2]
        I[1,1]  =  (mo*(W*W + H*H) - mi*(Wi*Wi + Hi*Hi))/12
        I[2,2]  =  (mo*(L*L + H*H) - mi*(L*L + Hi*Hi))/12
        I[3,3]  =  (mo*(L*L + W*W) - mi*(L*L + Wi*Wi))/12
        I[1,2]  =  zero
        I[1,3]  =  zero
        I[2,1]  =  zero
        I[2,3]  =  zero
        I[3,1]  =  zero
        I[3,2]  =  zero
        mo  =  (((1000*rho)*L)*W)*H
        mi  =  (((1000*rho)*L)*Wi)*Hi
        m  =  mo - mi
        r0b[1]  =  r0a[1] + Sa[1,1]*r[1] + Sa[1,2]*r[2] + Sa[1,3]*r[3]
        r0b[2]  =  r0a[2] + Sa[2,1]*r[1] + Sa[2,2]*r[2] + Sa[2,3]*r[3]
        r0b[3]  =  r0a[3] + Sa[3,1]*r[1] + Sa[3,2]*r[2] + Sa[3,3]*r[3]
        Ta[1]  = (I[1,1]*za[1]+I[1,2]*za[2]+I[1,3]*za[3])+pro3[1]+pro4[1]
        Ta[2]  =  (I[2,1]*za[1]+I[2,2]*za[2]+I[2,3]*za[3])+pro3[2]+pro4[2]
        Ta[3]  =  (I[3,1]*za[1]+I[3,2]*za[2]+I[3,3]*za[3])+pro3[3]+pro4[3]
```

```
        EXPAND_BLOCK ( i  IN  1,3 )
                wb[i]  =  wa[i]
                zb[i]  =  za[i]
                Fa[i]  =  m*(aa[i] + pro1[i]) + pro2[i]
                vb[i]  =  va[i] + vaux[i]
                ab[i]  =   aa[i] + prod1[i] + prod2[i]
                Fb[i]  =  Fa[i]
                Tb[i]  =  Ta[i] - prod3[i]
                rCM[i]  =  r0[i] + 1/2*LengthDir[i]
                EXPAND ( j  IN  1,3 )
                Sb[i,j]  =  Sa[i,j]
        END EXPAND_BLOCK
END COMPONENT
```

El último componente creado se denomina **Revolute**, este componente hereda las propiedades del componente abstracto **TwoTreeFrames.** El componente **Revolute** es un tipo de junta de revolución que dispone de un puerto de salida (Frame_B) y otro de entrada (Frame_A), donde Frame_B gira alrededor del eje n que se fija en Frame_A. Esta junta tiene 1 grado de libertad de rotación (véase figura 4.6) [9].

Los dos Frames coinciden cuando el ángulo de rotación es “phi=0”.

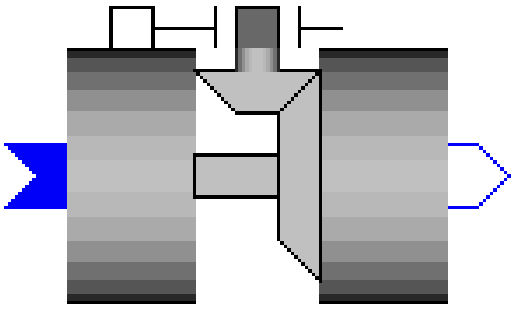

Figura 4.6. Componente Revolute

Estas son las variables principales que se declaran en el componente **Revolute**:

- **n:** eje de rotación resuelto en Frame_A.
- **q:** ángulo de rotación de compensación.

A continuación, se escribe el código desarrollado del componente **Revolute**:

```
COMPONENT  Revolute  IS_A  TwoTreeFrames
    PORTS
        IN  Flange  axis
        OUT  Flange  bearing
    DATA
        REAL  n[3] = {0,0,1}
        REAL  nn[3] = {0,0,1}
        REAL  pi = 3.14
        REAL  q0 = 0      "(rad)"
        REAL  r_rela[3] =  {0,0,0}     "(m)"
        REAL  v_rela[3] =  {0,0,0}     "(m/s)"
        REAL  a_rela[3] =  {0,0,0}     "(m/s2)"
    DECLS
        REAL  q       "(rad)"
        REAL  qd      "(rad/s)"
        REAL  qq      "(rad)"
        REAL  qdd     "(rad/s2)"
        REAL  sinq
        REAL  cosq
        REAL  S_rel[3,3]
        REAL  w_rela[3]    "(rad/s)"
        REAL  z_rela[3]    "(rad/s2)"
        REAL  pro[3]
    CONTINUOUS
        axis.phi = q
        bearing.phi = 0
        qd = q'
        qdd = qd'
        qq  =  q - (q0*pi)/180
        sinq  =  sin(qq)
        cosq  =  cos(qq)
        S_rel[1,1]  =  nn[1]*nn[1] + (1 - nn[1]*nn[1])*cosq
        S_rel[1,2]  =  nn[1]*nn[2] + (0 - nn[1]*nn[2])*cosq + nn[3]*sinq
        S_rel[1,3]  =  nn[1]*nn[3] + (0 - nn[1]*nn[3])*cosq - nn[2]*sinq
        S_rel[2,1]  =  nn[2]*nn[1] + (0 - nn[2]*nn[1])*cosq - nn[3]*sinq
        S_rel[2,2]  =  nn[2]*nn[2] + (1 - nn[2]*nn[2])*cosq
        S_rel[2,3]  =  nn[2]*nn[3] + (0 - nn[2]*nn[3])*cosq + nn[1]*sinq
        S_rel[3,1]  =  nn[3]*nn[1] + (0 - nn[3]*nn[1])*cosq + nn[2]*sinq
        S_rel[3,2]  =  nn[3]*nn[2] + (0 - nn[3]*nn[2])*cosq - nn[1]*sinq
        S_rel[3,3]  =  nn[3]*nn[3] + (1 - nn[3]*nn[3])*cosq
        EXPAND_BLOCK ( j  IN  1,3 )
            w_rela[j]  =  nn[j]*qd
            z_rela[j]  =  nn[j]*qdd
            r0b[j]  =  r0a[j]
```

```
        END EXPAND_BLOCK
        Sb[1,1] = Sa[1,1]*S_rel[1,1] + Sa[1,2]*S_rel[1,2] + Sa[1,3]*S_rel[1,3]
        Sb[1,2] = Sa[1,1]*S_rel[2,1] + Sa[1,2]*S_rel[2,2] + Sa[1,3]*S_rel[2,3]
        Sb[1,3] = Sa[1,1]*S_rel[3,1] + Sa[1,2]*S_rel[3,2] + Sa[1,3]*S_rel[3,3]
        Sb[2,1] = Sa[2,1]*S_rel[1,1] + Sa[2,2]*S_rel[1,2] + Sa[2,3]*S_rel[1,3]
        Sb[2,2] = Sa[2,1]*S_rel[2,1] + Sa[2,2]*S_rel[2,2] + Sa[2,3]*S_rel[2,3]
        Sb[2,3] = Sa[2,1]*S_rel[3,1] + Sa[2,2]*S_rel[3,2] + Sa[2,3]*S_rel[3,3]
        Sb[3,1] = Sa[3,1]*S_rel[1,1] + Sa[3,2]*S_rel[1,2] + Sa[3,3]*S_rel[1,3]
        Sb[3,2] = Sa[3,1]*S_rel[2,1] + Sa[3,2]*S_rel[2,2] + Sa[3,3]*S_rel[2,3]
        Sb[3,3] = Sa[3,1]*S_rel[3,1] + Sa[3,2]*S_rel[3,2] + Sa[3,3]*S_rel[3,3]
        vb[1] = S_rel[1,1]*va[1] + S_rel[1,2]*va[2] + S_rel[1,3]*va[3]
        vb[2] = S_rel[2,1]*va[1] + S_rel[2,2]*va[2] + S_rel[2,3]*va[3]
        vb[3] = S_rel[3,1]*va[1] + S_rel[3,2]*va[2] + S_rel[3,3]*va[3]
        wb[1] = S_rel[1,1]*(wa[1] + w_rela[1]) + S_rel[1,2]*(wa[2] +
        w_rela[2]) + S_rel[1,3]*(wa[3] + w_rela[3])
        wb[2] = S_rel[2,1]*(wa[1] + w_rela[1]) + S_rel[2,2]*(wa[2] +
        w_rela[2]) + S_rel[2,3]*(wa[3] + w_rela[3])
        wb[3] = S_rel[3,1]*(wa[1] + w_rela[1]) + S_rel[3,2]*(wa[2] +
        w_rela[2]) + S_rel[3,3]*(wa[3] + w_rela[3])
        ab[1] = S_rel[1,1]*aa[1] + S_rel[1,2]*aa[2] + S_rel[1,3]*aa[3]
        ab[2] = S_rel[2,1]*aa[1] + S_rel[2,2]*aa[2] + S_rel[2,3]*aa[3]
        ab[3] = S_rel[3,1]*aa[1] + S_rel[3,2]*aa[2] + S_rel[3,3]*aa[3]
        pro[1] = - wa[3]*w_rela[2] + wa[2]*w_rela[3]
        pro[2] = wa[3]*w_rela[1] - wa[1]*w_rela[3]
        pro[3] = - wa[2]*w_rela[1] + wa[1]*w_rela[2]
        zb[1] = S_rel[1,1]*(za[1] + z_rela[1]) + S_rel[1,2]*(za[2] +
        z_rela[2]) + S_rel[1,3]*(za[3] + z_rela[3]) + pro[1]
        zb[2] = S_rel[2,1]*(za[1] + z_rela[1]) + S_rel[2,2]*(za[2] +
        z_rela[2]) + S_rel[2,3]*(za[3] + z_rela[3]) + pro[2]
        zb[3] = S_rel[3,1]*(za[1] + z_rela[1]) + S_rel[3,2]*(za[2] +
        z_rela[2]) + S_rel[3,3]*(za[3] + z_rela[3]) + pro[3]
        Fa[1] = S_rel[1,1]*Fb[1] + S_rel[2,1]*Fb[2] + S_rel[3,1]*Fb[3]
        Fa[2] = S_rel[1,2]*Fb[1] + S_rel[2,2]*Fb[2] + S_rel[3,2]*Fb[3]
        Fa[3] = S_rel[1,3]*Fb[1] + S_rel[2,3]*Fb[2] + S_rel[3,3]*Fb[3]
        Ta[1] = S_rel[1,1]*Tb[1] + S_rel[2,1]*Tb[2] + S_rel[3,1]*Tb[3]
        Ta[2] = S_rel[1,2]*Tb[1] + S_rel[2,2]*Tb[2] + S_rel[3,2]*Tb[3]
        Ta[3] = S_rel[1,3]*Tb[1] + S_rel[2,3]*Tb[2] + S_rel[3,3]*Tb[3]
        axis.tau = nn[1]*Tb[1] + nn[2]*Tb[2] + nn[3]*Tb[3]
END COMPONENT
```

Capítulo 5

Resultados de la simulación

5.1 Experimento Pendulum con amortiguador

Para inicializar la programación multicuerpo con EcosimPro, se ha realizado un experimento llamado *Pendulum,* dicho experimento se ha tomado de Modelica para poder comparar los resultados obtenidos con EcosimPro.

El experimento *Pendulum* es un sistema físico constituido por hilo inextensible sujeto a un punto fijo del que cuelga una masa que puede oscilar libremente. Para pequeñas oscilaciones, la masa describe un movimiento armónico simple.

Se dice que un punto sigue un movimiento vibratorio armónico simple (m.a.s.) cuando su posición en función del tiempo es una sinusoide, es decir, varía según una ecuación de tipo senoidal o cosenoidal. Es un movimiento periódico de vaivén, en el que un cuerpo oscila a un lado y a otro de su posición de equilibrio en una dirección determinada y en intervalos iguales de tiempo (véase figura 5.1).

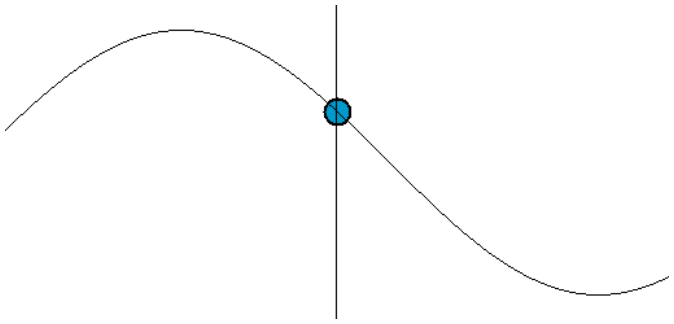

Figura 5.1. Movimiento vibratorio armónico simple

El m.a.s. es un movimiento acelerado no uniformemente. Su aceleración es proporcional al desplazamiento y de signo opuesto a este. Toma su valor máximo en los extremos de la trayectoria, mientras que es mínimo en el centro.

En la figura 5.2 se observa un diagrama de fuerzas de un péndulo simple. El péndulo describe una trayectoria circular, un arco de una circunferencia de radio *l*.

Las fuerzas que actúan sobre la partícula de masa *m* son dos:

- el peso *mg*.
- La tensión *T* del hilo.

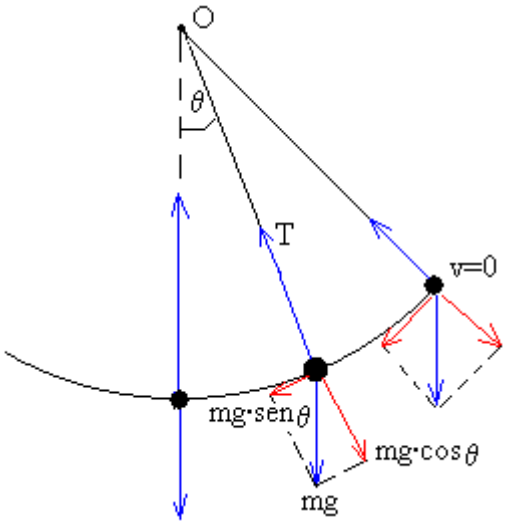

Figura 5.2. Diagrama de fuerzas del péndulo

La tensión de la cuerda no es constante, sino que varía con la posición angular θ. El valor máximo de la tensión se alcanza cuando $\theta=0$, donde el péndulo pasa por la posición de equilibrio (la velocidad es máxima). El valor mínimo de la tensión se alcanza cuando $\theta=\theta_0$ (la velocidad es nula).

En la posición $\theta=\theta_0$ el péndulo solamente tiene energía potencial, que se transforma en energía cinética cuando el péndulo pasa por la posición de equilibrio.

Para ver si los resultados obtenidos en la simulación con EcosimPro son correctos, se han comparado las gráficas de las variables obtenidas con las que se obtuvieron en el mismo experimento con el programa Modelica. En el apartado 5.2 se muestran estos resultados, como se manejan muchas variables en el sistema, sólo se van a mostrar las variables más representativas del experimento.

El experimento *Pendulum* que se ha tomado de Modelica y consta de cuatro elementos principales interconectados entre sí, un sistema inercial (inertial) conectado a una junta de revolución (revolute), esta junta está unida por un lado a un amortiguador (damper) y por otro lado a una masa (boxBody). En la figura 5.3 se puede observar el esquema de dicho experimento, y se aprecia claramente como se unen entre sí los puertos de los componentes.

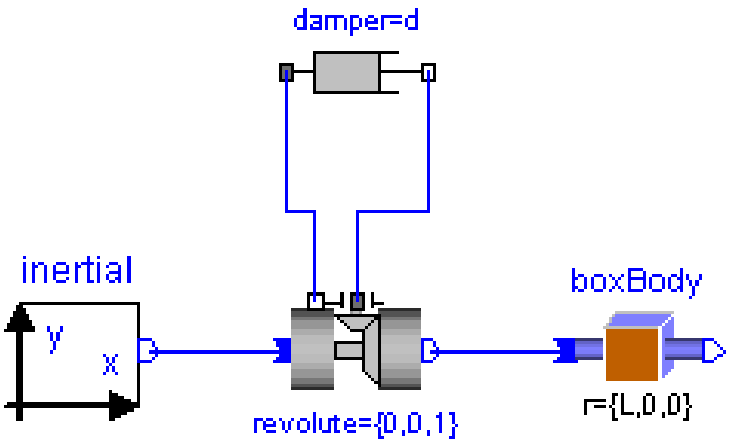

Figura 5.3. Esquema del experimento *Pendulum*

A continuación, se muestra el código del ejemplo *Pendulum* en EcosimPro, donde se instancian los cuatro componentes mencionados antes y posteriormente se conectan entre sí:

```
COMPONENT  Pendulum
        TOPOLOGY
                InertialSystem  inertial
                BoxBody  boxBody
                Revolute  revolute
                Damper  damper
                CONNECT  inertial.frame_B  TO  revolute.frame_A
                CONNECT  revolute.frame_B  TO  boxBody.frame_A
                CONNECT  damper.flange_B  TO  revolute.axis
                CONNECT  damper.flange_A  TO  revolute.bearing
END  COMPONENT
```

El siguiente código pertenece al experimento creado en EcosimPro donde se inicializan las variables que no lo han sido antes, y donde se especifica el tiempo y los intervalos de integración:

```
EXPERIMENT  exp1  ON  Pendulum.default
        DECLS
        INIT
                -- initial values for state variables
                revolute.qd  =  0
                damper.phi_rel  =  0
        BOUNDS
        BODY
                -- report results in file reportAll.rpt
                REPORT_TABLE ("reportAll.rpt", "*")
                TIME  =  0
                TSTOP  =  15
                CINT  =  0.1
                INTEG()
END EXPERIMENT
```

5.2 Resultados de la simulación del Pendulum con amortiguador

La primera variable a comparar es el ángulo "**q**" del componente **revolute**, se trata del ángulo que varía en el movimiento del péndulo. En la figura 5.4 se muestra el resultado en Modelica cuyo valor se encuentra entre *0 rad* y *-pi rad*, el ángulo va disminuyendo por el efecto del amortiguador. En la figura 5.5 se ve el mismo resultado en EcosimPro pero con el signo contrario, esto se debe a la diferencia entre los sistemas de referencia.

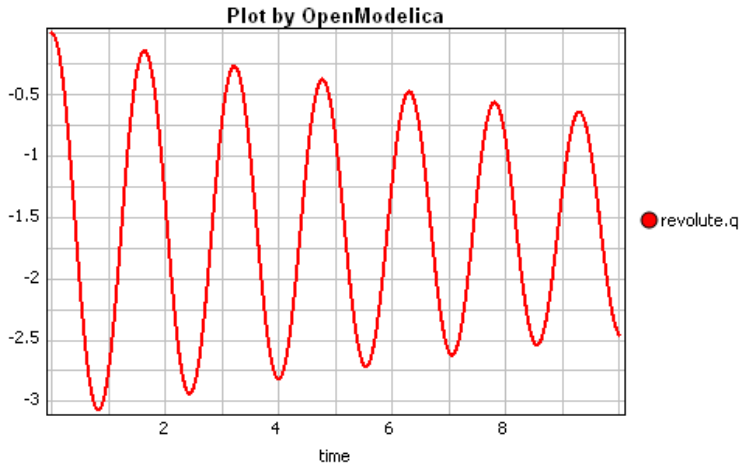

Figura 5.4. Gráfica de Revolute q (Modelica)

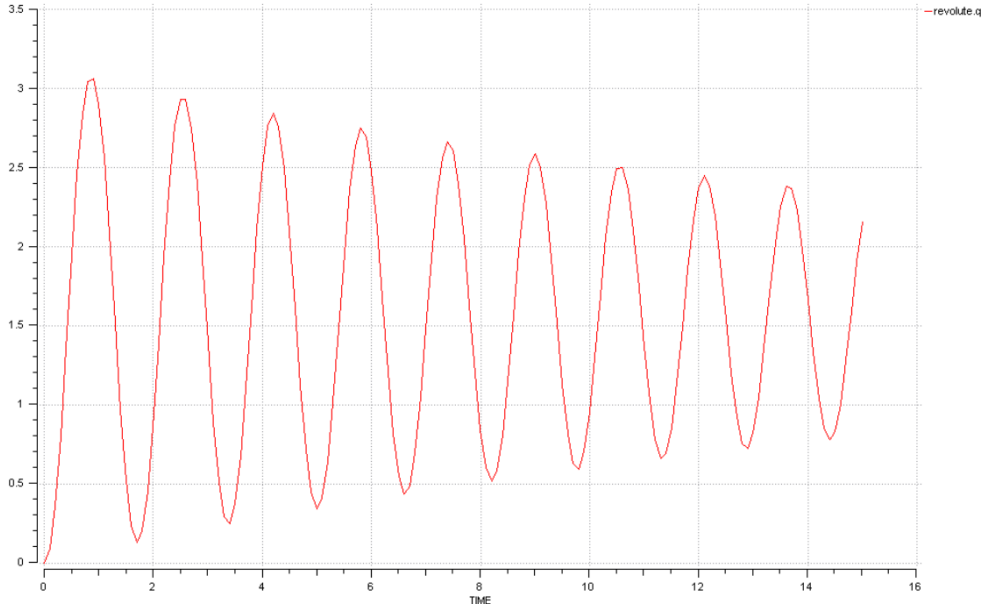

Figura 5.5. Gráfica de Revolute q (EcosimPro)

La siguiente variable a comparar es la velocidad angular "**qd**" del componente **revolute**, se calcula derivando el angulo "q". En la figura 5.6 se muestra el resultado en Modelica cuyo valor se encuentra entre *-6 rad/s* y *6rad/s*. La velocidad angular, igual que el ángulo anterior, va disminuyendo por el efecto del amortiguador. En la figura 5.7 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

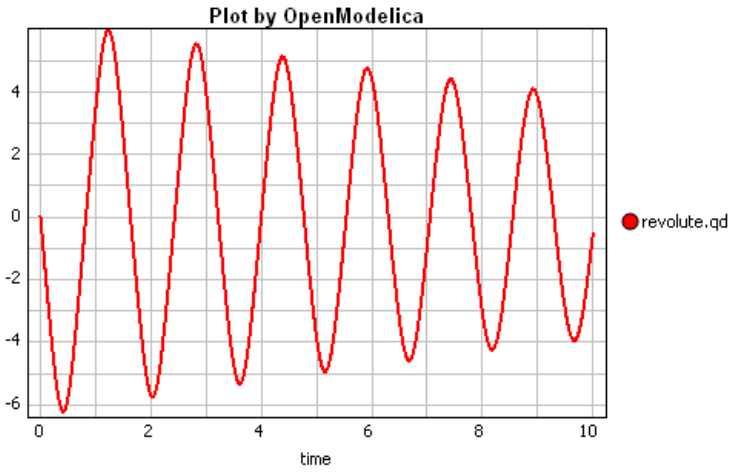

Figura 5.6. Gráfica de Revolute qd (Modelica)

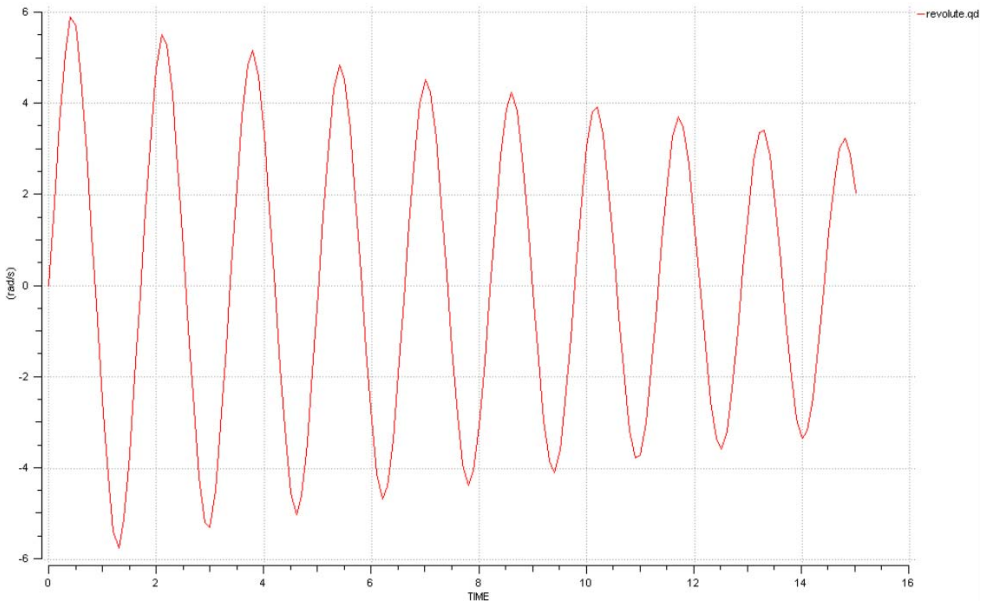

Figura 5.7. Gráfica de Revolute qd (EcosimPro)

La siguiente variable que se compara es el par "**Ta**" a la entrada del componente **revolute**. En la figura 5.8 se muestra el resultado en Modelica cuyo valor se encuentra entre *-6 Nm* y *6 Nm*, el valor del par al igual que el ángulo, va disminuyendo por el efecto del amortiguador que absorbe energía del sistema. En la figura 5.9 se observa el mismo resultado en EcosimPro.

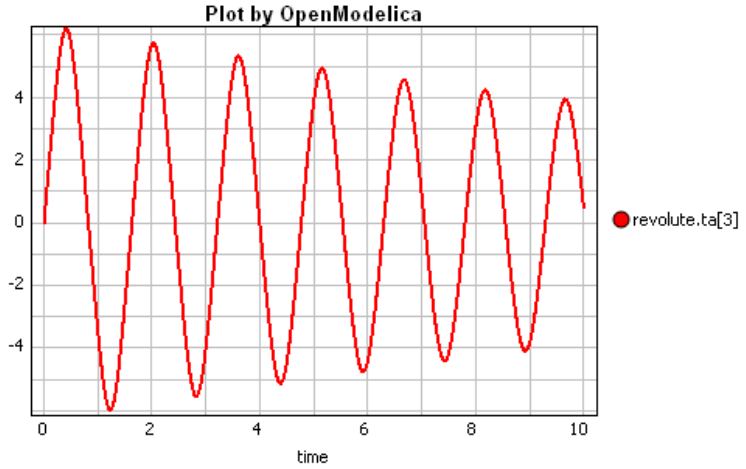

Figura 5.8. Gráfica de Revolute Ta (Modelica)

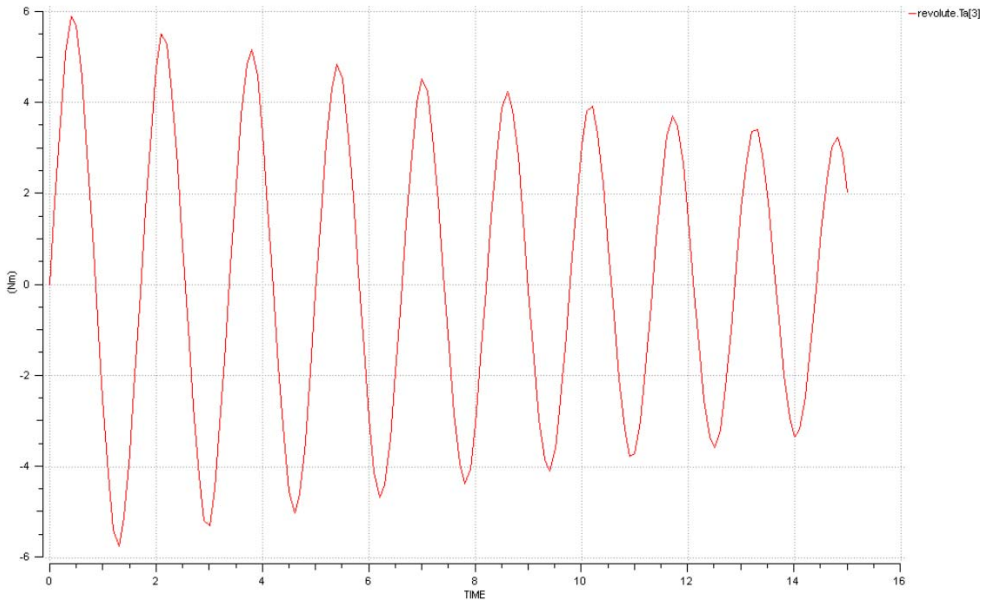

Figura 5.9. Gráfica de Revolute Ta (EcosimPro)

La siguiente variable a comparar es el par "**Tb**" a la salida del componente **revolute**. Se puede observar que el par al inicio de la junta de revolución (Ta) coincide con el par al final de esta. En la figura 5.10 se muestra el resultado en Modelica cuyo valor se encuentra entre *-6 Nm* y *6 Nm*, el valor del par al igual que el ángulo. En la figura 5.11 se observa el mismo resultado en EcosimPro.

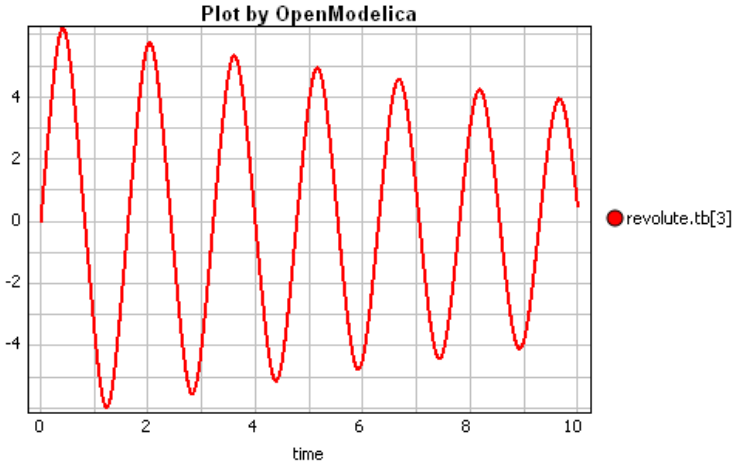

Figura 5.10. Gráfica de Revolute Tb (Modelica)

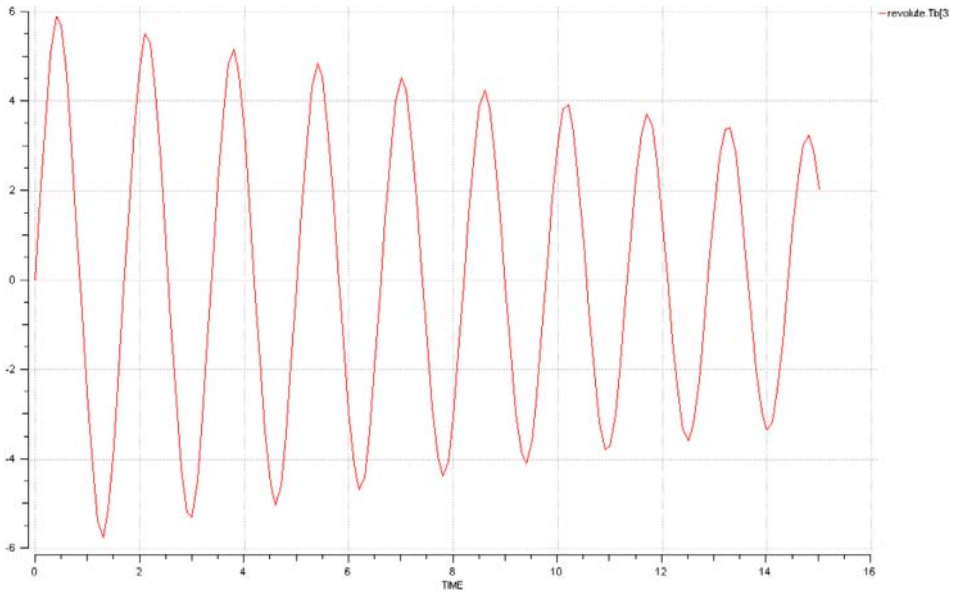

Figura 5.11. Gráfica de Revolute Tb (EcosimPro)

En la figuras 5.12 y 5.13 se muestra la evolución de la variable velocidad angular "**w_rela**" del componente **revolute**, a lo largo del tiempo, para Modelica y EcosimPro. El valor del resultado obtenido en Modelica se encuentra entre *-6 rad/s* y *6rad/s*. El valor del resultado obtenido en EcosimPro es el mismo pero con el inicio del movimiento en sentido contrario.

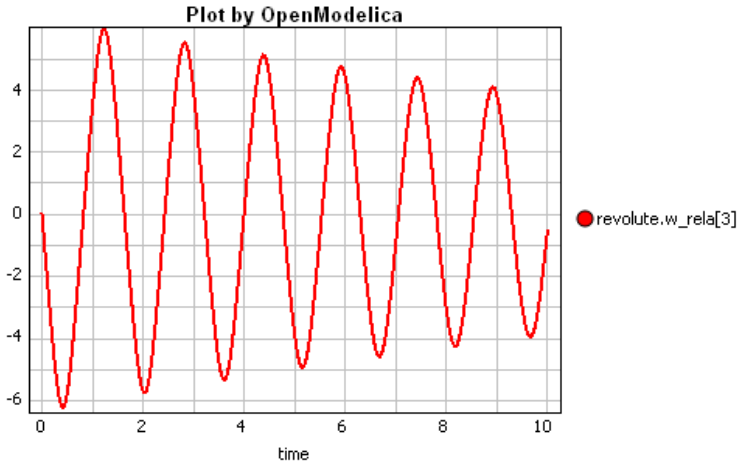

Figura 5.12. Gráfica de Revolute w_rela (Modelica)

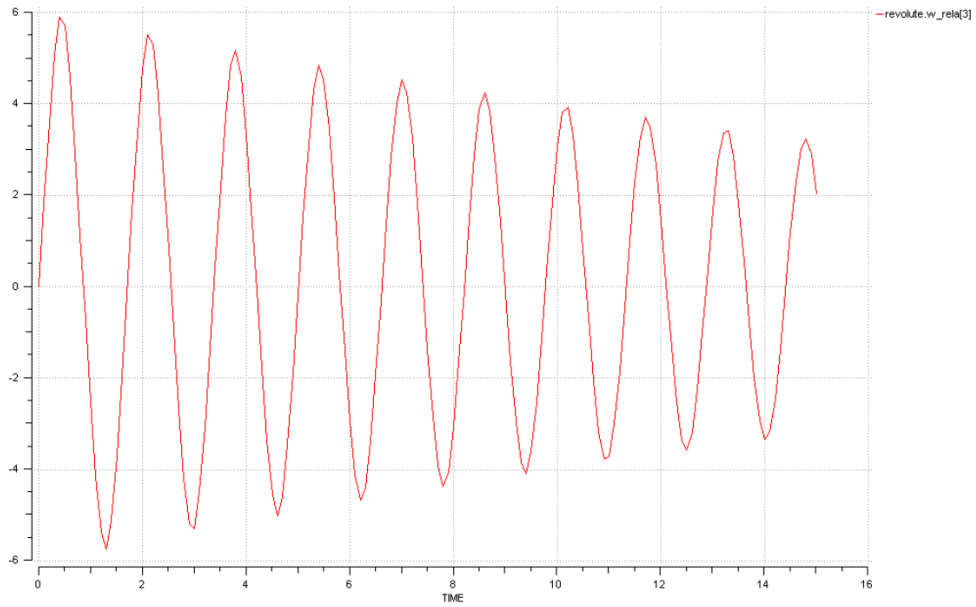

Figura 5.13. Gráfica de Revolute w_rela (EcosimPro)

La siguiente variable a comparar es la aceleración angular "**z_rela**" de la junta de revolución **revolute**, se calcula derivando la velocidad angular "qd". En la figura 5.14 se muestra el resultado en Modelica cuyo valor se encuentra entre *-20 rad/s²* y *20 rad/s²*, la aceleración, igual que la velocidad, va disminuyendo por el efecto del amortiguador. En la figura 5.15 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

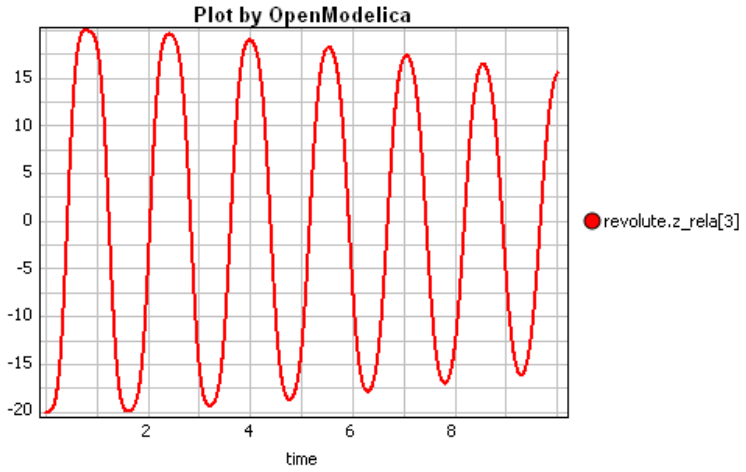

Figura 5.14. Gráfica de Revolute z_rela (Modelica)

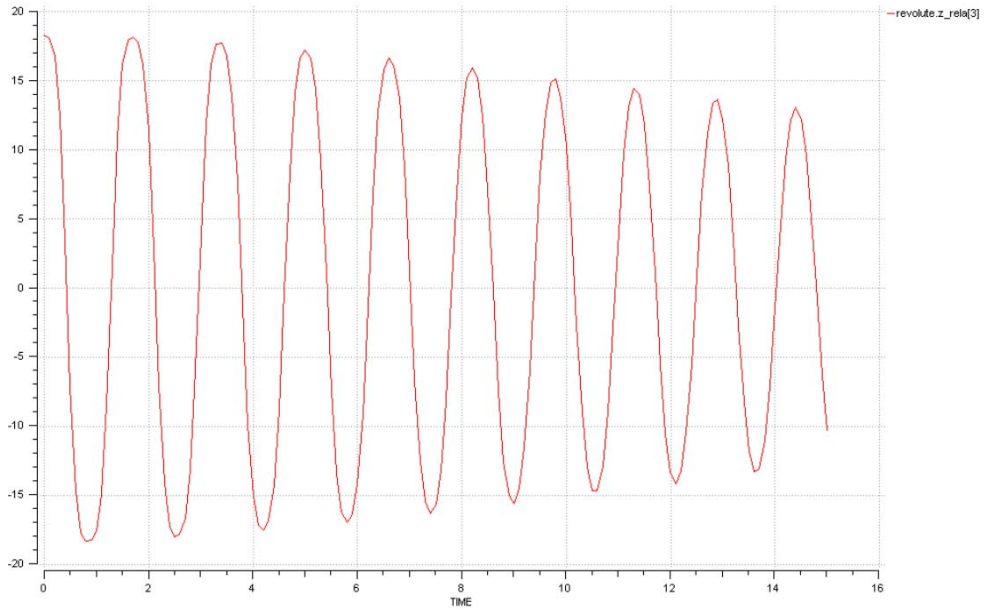

Figura 5.15. Gráfica de Revolute z_rela (EcosimPro)

La siguiente variable que se compara es el par en el puerto de entrada "**frame_A.T**" del componente **boxBody**. En la figura 5.16 se muestra el resultado en Modelica cuyo valor se encuentra entre *-6 Nm* y *6 Nm*, el valor va disminuyendo por el efecto del amortiguador que absorbe energía del sistema. Este par coincide con el obtenido al final del componente revolute (Tb). En la figura 5.17 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

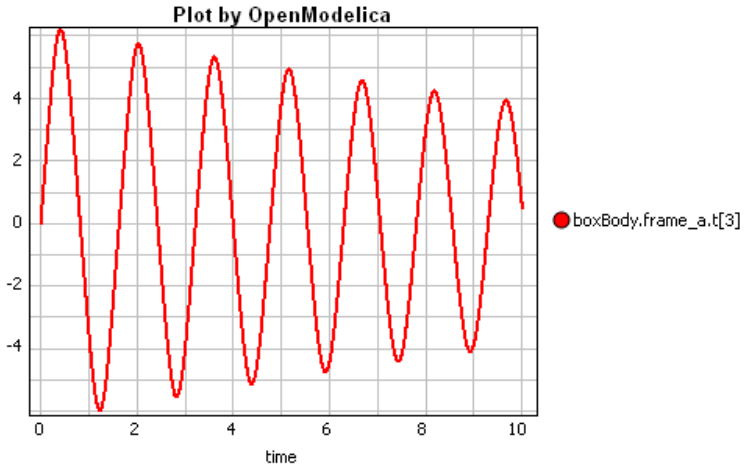

Figura 5.16. Gráfica de BoxBody frame_A T (Modelica)

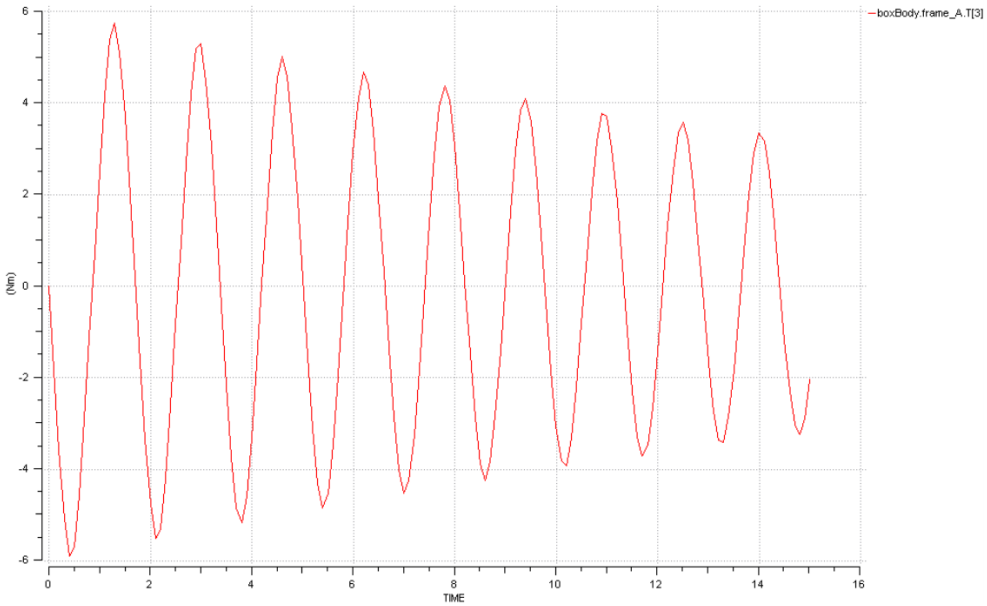

Figura 5.17. Gráfica de BoxBody frame_A T (EcosimPro)

La siguiente variable a comparar es el par en el puerto de entrada "**flange_A.tau**" del componente **damper**. En la figura 5.18 se muestra el resultado en Modelica cuyo valor se encuentra entre *-6 Nm* y *6 Nm*. En la figura 5.19 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

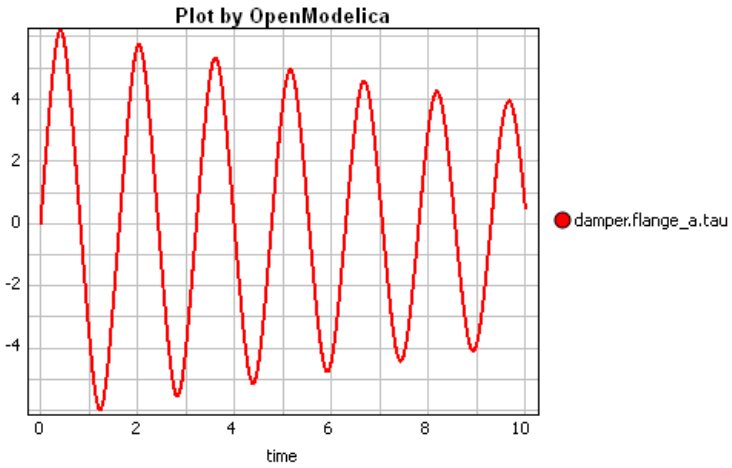

Figura 5.18. Gráfica de Damper flange_A tau (Modelica)

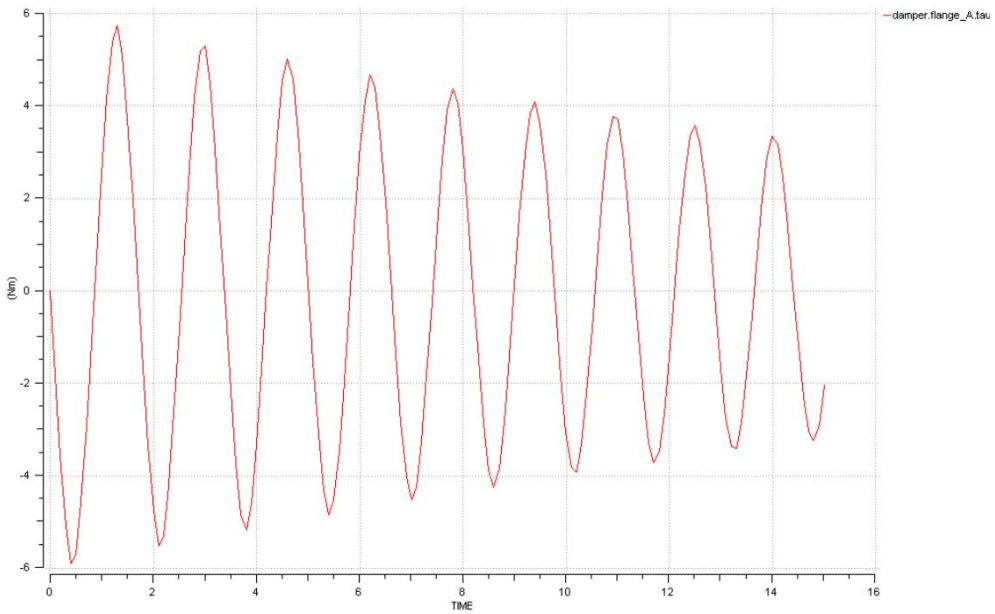

Figura 5.19. Gráfica de Damper flange_A tau (EcosimPro)

La siguiente variable que se compara es el par en el puerto de salida "**flange_B.tau**" del componente **damper**. En la figura 5.20 se muestra el resultado en Modelica cuyo valor se encuentra entre *-6 Nm* y *6 Nm*. En la figura 5.21 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

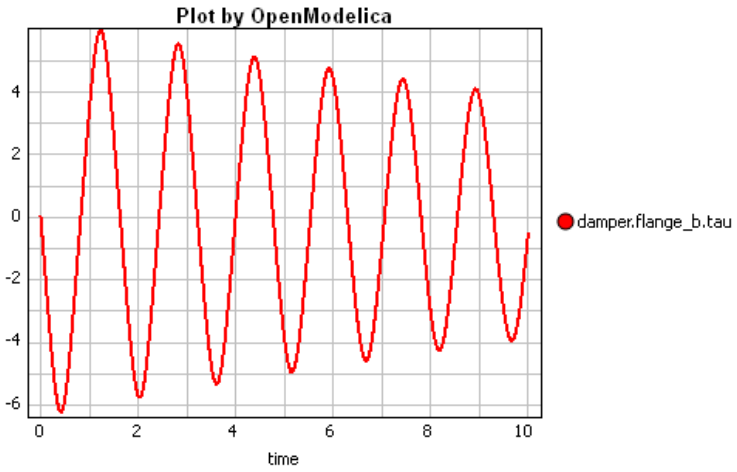

Figura 5.20. Gráfica de Damper flange_B tau (Modelica)

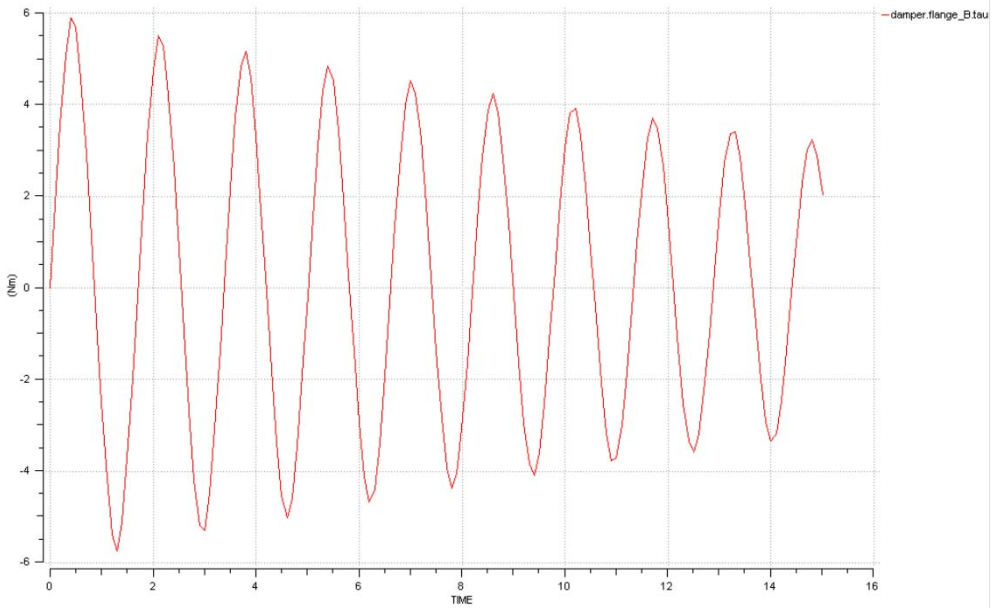

Figura 5.21. Gráfica de Damper flange_B tau (EcosimPro)

La siguiente variable a comparar es el ángulo "**flange_B.phi**" en el puerto de salida del componente **damper**. En la figura 5.22 se muestra el resultado en Modelica cuyo valor se encuentra entre *0 rad* y *-pi rad*, el ángulo va disminuyendo por el efecto del amortiguador. En la figura 5.23 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

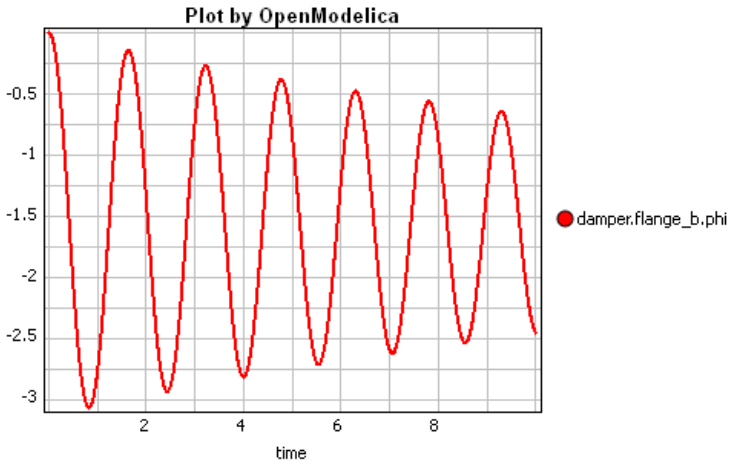

Figura 5.22. Gráfica de Damper flange_B phi (Modelica)

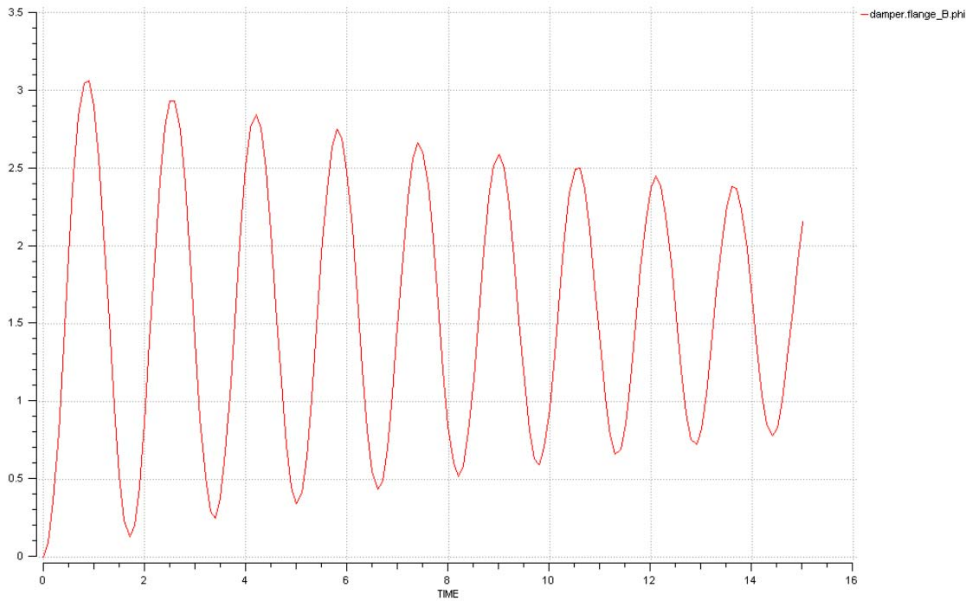

Figura 5.23. Gráfica de Damper flange_B phi (EcosimPro)

La siguiente variable que se compara es el ángulo relativo "**phi_rel**" del componente **damper**. En la figura 5.24 se muestra el resultado en Modelica cuyo valor se encuentra entre *0 rad* y *-pi rad*, en el experimento realizado este ángulo coincide con el ángulo "phi" de salida del componente damper. En la figura 5.25 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

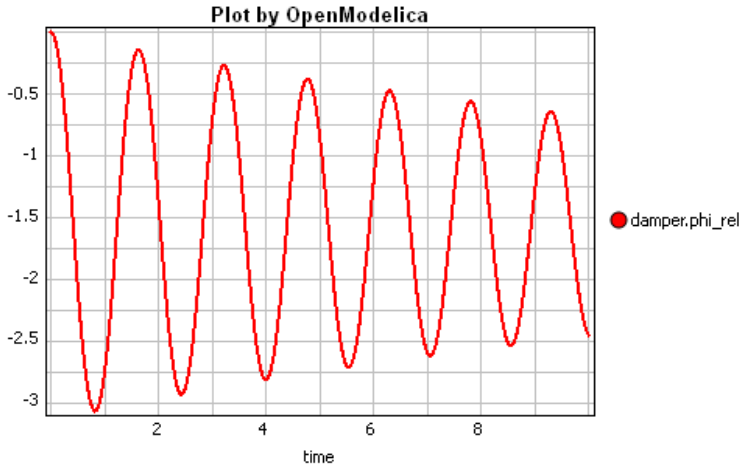

Figura 5.24. Gráfica de Damper phi_rel (Modelica)

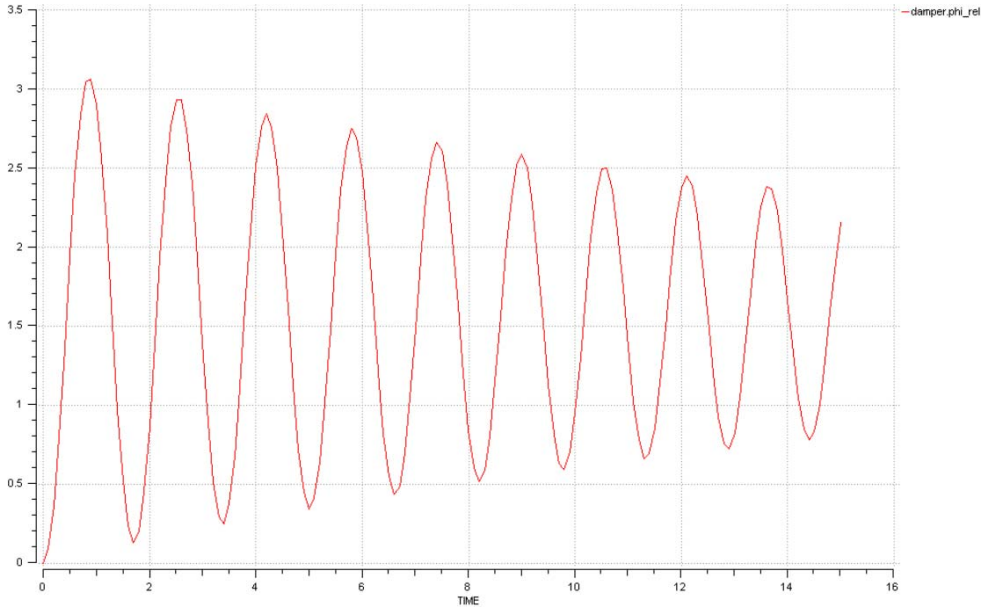

Figura 5.25. Gráfica de Damper phi_rel (EcosimPro)

La siguiente variable a comparar es la velocidad angular "**w_rel**" del componente **damper**, se calcula mediante la derivada del ángulo "phi_rel". En la figura 5.26 se muestra el resultado en Modelica cuyo valor se encuentra entre *-6 rad/s* y *6rad/s*. En la figura 5.27 se ve el mismo resultado en EcosimPro pero con el inicio del movimiento en sentido contrario.

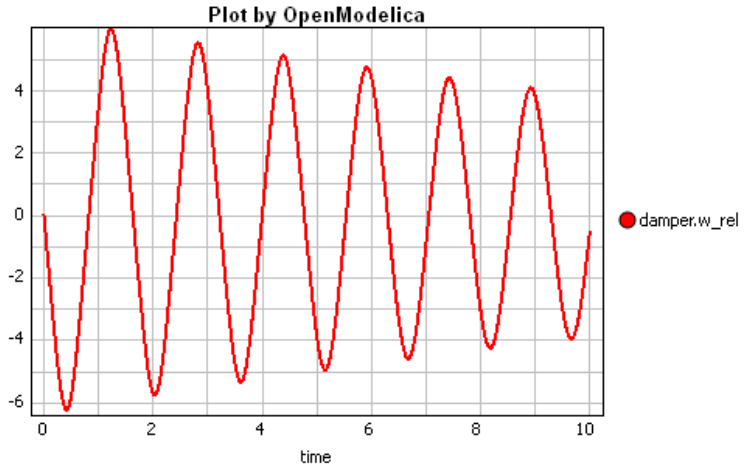

Figura 5.26. Gráfica de Damper w_rel (Modelica)

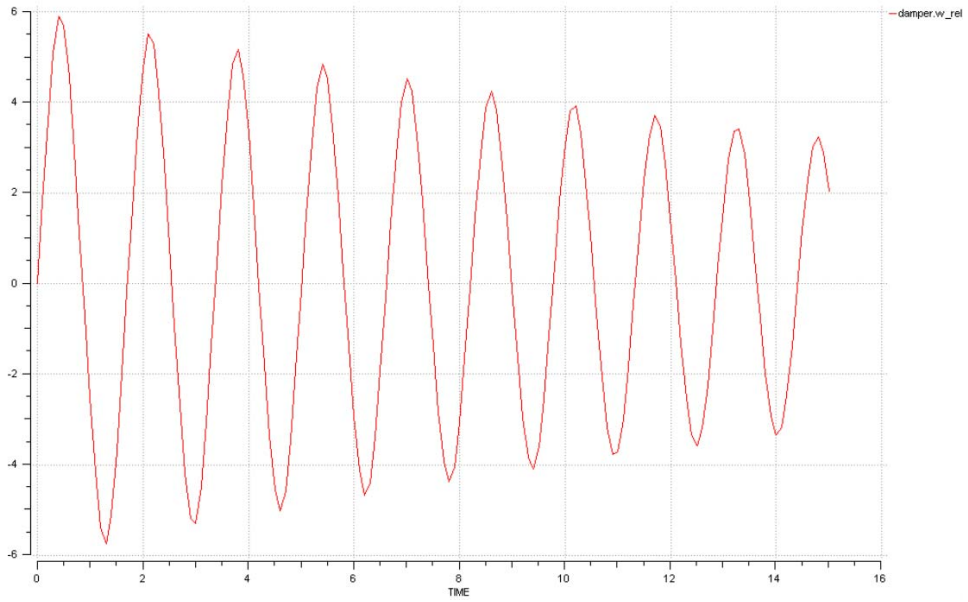

Figura 5.27. Gráfica de Damper w_rel (EcosimPro)

5.3 Experimento Pendulum sin amortiguador

En el siguiente experimento, se estudia el comportamiento de un péndulo simple sin ningún elemento que quite energía al sistema. Con este experimento se comparan los resultados obtenidos en EcosimPro con los del experimento anterior (con amortiguador), y así comprobar el efecto que producía el amortiguador en el sistema. En este caso se ha modelado un sistema igual que el del punto 5.1 pero eliminando el amortiguador (véase figura 5.28).

El amortiguador es un dispositivo que absorbe energía, utilizado normalmente para disminuir las oscilaciones no deseadas de un movimiento periódico como el que nos ocupa.

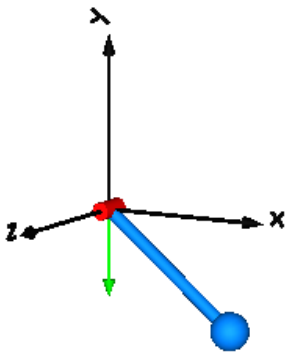

Figura 5.28. Péndulo simple

A continuación, se muestra el código del nuevo experimento:

```
COMPONENT Pendulum2
        TOPOLOGY
                InertialSystem inertial
                BoxBody boxBody
                Revolute revolute
                CONNECT inertial.frame_B TO revolute.frame_A
                CONNECT revolute.frame_B TO boxBody.frame_A
END COMPONENT
```

5.4 Resultados de la simulación del Pendulum sin amorigurador

En la figura 5.29 se observa el resultado de la variable ángulo "**q**" del componente **revolute** en EcosimPro. En este caso, al no tener amortiguador, el ángulo del péndulo oscila prácticamente constante hasta que alcanza el equilibrio y a partir de ese momento la oscilación es constante.

El resultado obtenido varía entre *0 rad* y *pi rad,* se comporta como un péndulo ideal, con esta variable se aprecia claramente el efecto que produce en el amortiguador en el sistema (véase figura 5.30).

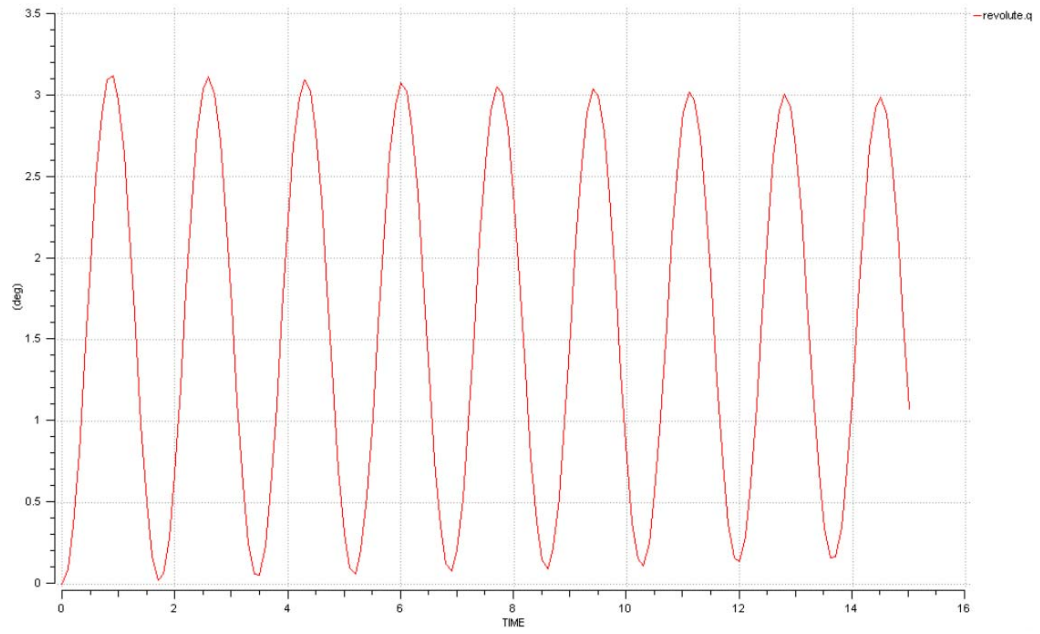

Figura 5.29. Gráfica de Revolute q sin amortiguador

En la figura 5.30 se muestra el resultado de la variable "**q**" del experimento con amortiguador obtenido en el apartado 5.2.

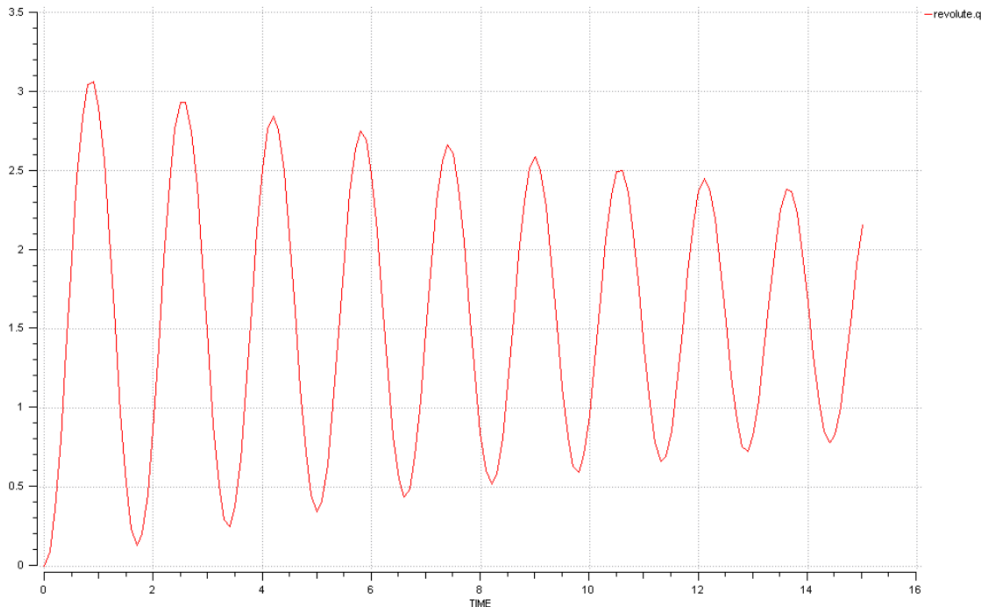

Figura 5.30. Gráfica de Revolute q con amortiguador

En la figura 5.31 se observa el resultado de la variable velocidad angular "**w_rela**" del componente **revolute** en EcosimPro. En este caso, al no tener amortiguador, la velocidad angular oscila constante durante el experimento. En la figura 5.32 se muestra el resultado de la misma variable pero con el efecto del amortiguador, se observa claramente la pérdida de energía del sistema en este caso.

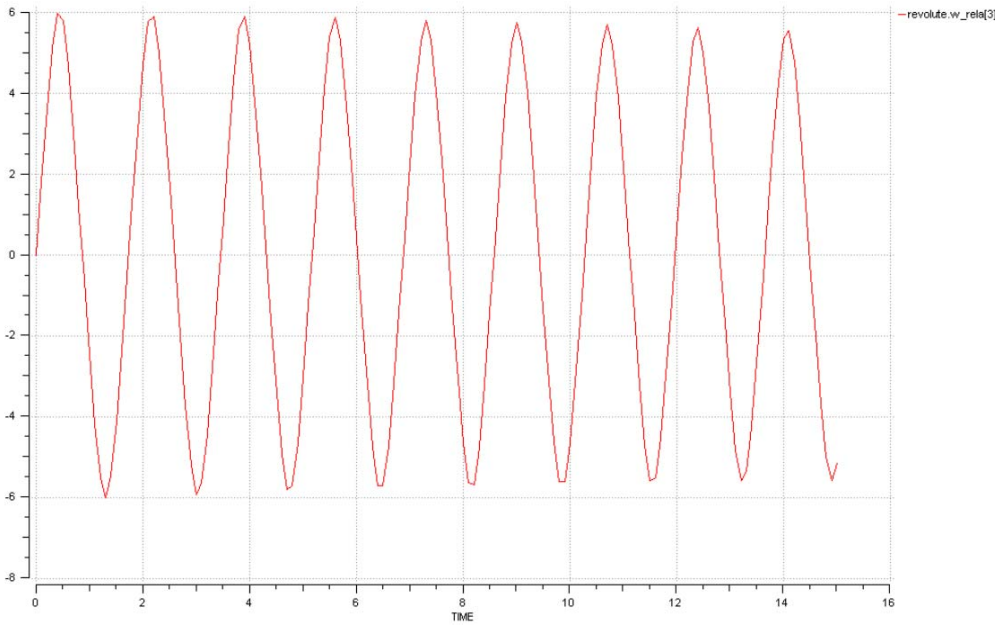

Figura 5.31. Gráfica de Revolute w_rela sin amortiguador

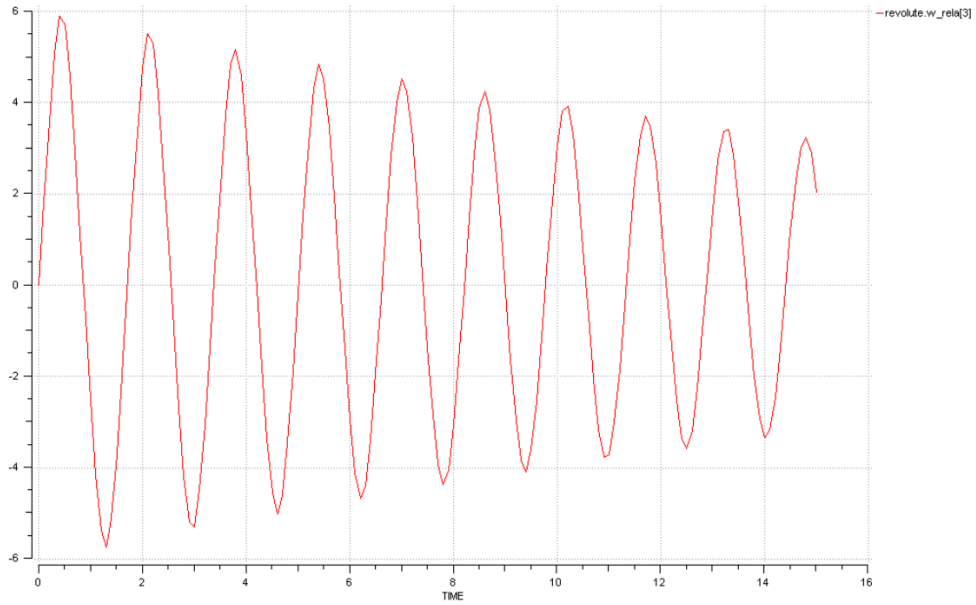

Figura 5.32. Gráfica de Revolute w_rela con amortiguador

Capítulo 6

Conclusiones y desarrollos futuros

6.1 Conclusiones

En este proyecto, se ha realizado un experimento para la iniciación al modelado de sistemas multicuerpo con EcosimPro. En dicho experimento se han forjado las bases para este tipo de programación y a partir de aquí poder programar sistemas más complejos. Esto es importante ya que no existía ninguna base para dicha programación con esta herramienta.

Uno de los mayores inconvenientes encontrados ha sido el averiguar que EcosimPro no contempla el tratamiento matricial de las ecuaciones, complicando en gran medida el proceso de modelado. La mayor ventaja de la estructura matricial es la sencillez y la claridad de las expresiones resultantes. Para la programación multicuerpo con este programa, es muy importante tener claros los flujos de las variables que se unen en los distintos puertos.

Como conclusión final del proyecto se puede decir que los resultados obtenidos son satisfactorios, se ha conseguido que funcione la simulación del sistema multicuerpo con EcosimPro obteniendo los mismos resultados que con el programa Modelica. Aunque es verdad, que el código resultante no es todo lo eficiente que hubiéramos deseado por las particularidades de este programa, ya qué tiene demasiadas líneas de código.

En este momento, pese a las mayores capacidades de cálculo que ofrece EcosimPro, personalmente elegiría Modelica para la programación multicuerpo. Modelica permite trabajar con la mayoría de las operaciones matriciales, facilitando en gran medida al usuario la programación. EcosimPro no está diseñado para la programación multicuerpo, y esto dificulta su utilización en este campo.

6.2 Desarrollos futuros

A partir de las conclusiones y resultados extraídos de este Proyecto Fin de Carrera se pueden proponer una serie de futuras líneas de trabajo.

Lo primero sería solucionar el tratamiento matricial de las ecuaciones en EcosimPro, esto se podría solucionar creando una librería dedicada exclusivamente al álgebra matricial con todas las operaciones necesarias. Aún así, EcosimPro no está diseñado para realizar operaciones con vectores y matrices, por tanto, la programación siempre va a ser bastante larga y complicada.

Una vez solucionado el problema del tratamiento matricial con EcosimPro, hay muchas posibilidades de evolucionar la librería multicuerpo del programa:

- Crear una librería de juntas cinemáticas.
- Crear una librería de elementos de fuerza como amortiguadores o muelles.
- Crear una librería de puertos más extensa.
- Crear una librería con distintos tipos de cuerpos.
- Crear una librería de sensores.

Con estas librerías completas y los problemas descritos solucionados, EcosimPro podría llegar a ser una herramienta útil para la programación multicuerpo.

Capítulo 7

Bibliografía

[1] Shabana, A. A."Dynamics of multibody systems", 2nd ed., Cambridge University Press, 2005

[2] OpenCourseWare. "Análisis de sistemas multicuerpo". Universidad Politécnica de Madrid. 2005. http://ocw.upm.es/ingenieria-mecanica/simulacion-en-ingenieria-mecanica/contenidos/teoria/, accedido en junio 2011

[3] Peter Fritzson. "Principles of Object-Oriented Modelling and Simulation with Modelica 2.1", IEEE Press, 2004

[4] Garcia de Jalón, J. and Bayo E. "Kinematic and Dynamic Simulation of Multibody Systems". Springer Verlag, 1994.

[5] Agulló, J., "Mecánica de la partícula y del sólido rígido", OK Punt, 1996

[6] F.Vázquez, J.Jimenez. "Introducción al Modelado y Simulación con EcosimPro", Pearson, 2010.

[7] Jesús Vidal. "Un método general, sencillo y eficiente, para la definición y simulación numérica de sistemas multicuerpo". Sección de publicaciones de la ETS de Ingenieros Industriales de Madrid, 2006.

[8] EcosimPro/EL Modelling Language (versión 4.6, 2009)

[9] http://www.ida.liu.se/~pelab/realsim/library/ModelicaAdditions/docu/ModelicaAdditions_MultiBody.html#ModelicaAdditions.MultiBody, accedido en septiembre 2011